IF THESE WALLS COULD TALK: AUTOMATED PERFORMANCE MEASUREMENT FOR BUILDING MODELING DECISIONS USING DATA ANALYTICS

A Dissertation
Presented to
The Academic Faculty

By

Saman Yarmohammadi

In Partial Fulfillment
of the Requirement Degree
Doctor of Philosophy in the
School of Building Construction

Georgia Institute of Technology
May 2018

Copyright © 2018 by Saman Yarmohammadi

IF THESE WALLS COULD TALK: AUTOMATED PERFORMANCE MEASUREMENT FOR BUILDING MODELING DECISIONS USING DATA ANALYTICS

Approved by:

Dr. Daniel Castro-Lacouture, Advisor
School of Building Construction
Georgia Institute of Technology

Dr. Eunhwa Yang
School of Building Construction
Georgia Institute of Technology

Dr. Kamran Paynabar
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. John Haymaker, P.E.
Director of Research
Perkins+Will Design Company

Dr. Daniel Baerlecken
School of Architecture
Georgia Institute of Technology

Date Approved: November 20th, 2017

To my beloved wife,

Paran

To my kind mom and dad,

Sedigheh and *Alim*

ACKNOWLEDGMENTS

I am most grateful to my advisor, Dr. Daniel Castro-Lacouture, for his exquisite attention to detail and immense knowledge. The completion of my dissertation would have not been possible without his sincere advice and kind support. Special thanks to Dr. John Haymaker who provided me with invaluable comments and collaboration during my doctoral study. I would also like to extend my deepest appreciation to Dr. Kamran Paynabar for his continuous support and mentorship throughout this multi-disciplinary research. Furthermore, I thank my other committee members, Dr. Eunhwa Yang and Dr. Daniel Baerlecken, for sharing their helpful recommendations and insights into this study.

I cannot begin to express my thanks to my dear friends. I am deeply indebted to Dr. Kia Mostaan, Dr. Mohammad Ilbeigi, Dr. Reza Pourabolghasem, Mr. Mostafa Reisi, Mrs. Arezoo Shirazi, Ms. Shiva Bahrami, and Mr. Minsoo Baek. Their generosity in sharing their indispensable experience and expertise made me a better researcher.

There are people in everyone's lives who make success both possible and rewarding. I am extremely grateful to my wife, Paran, for being my rock throughout this endeavor. I would also like to thank my loving parents, Sedigheh and Alim, and my supportive siblings, Sadegh, Solmaz, and Siamak, for being my source of strength and inspiration. I am truly blessed for having you in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AEC	Architectural, Engineering, and Construction
AIA	American Institute of Architects
API	Application Programming Interface
BIM	Building Information Modeling
CII	Construction Industry Institute
CPM	Critical Path Method
CSV	Comma Separated Values
DFS	Depth-First-Search
EDD	Earliest Due Date
ENR	Engineering News-Record
GSP	Generalized Sequential Pattern
GST	Generalized Suffix Tree
GUID	Globally Unique Identifier
IFC	Issued for Construction
KDD	Knowledge Discovery in Databases
LoD	Level of Development
OOP	Object Oriented Programming
RFI	Request For Information

SPM	Sequential Pattern Mining
UI	User Interface
VDC	Virtual Design and Construction

SUMMARY

Building information modeling (BIM) is instrumental in documenting design, enhancing customer experience, and improving product functionality in capital projects. However, high-quality building models do not happen by accident, but rather because of a managed process that involves several participants from different disciplines and backgrounds. Throughout this process, the different priorities of design modelers often result in conflicts that can negatively impact project outcomes. There is a need for effective management of the modeling process to prevent such unwanted outcomes. Effective management of this process requires an ability to closely monitor the modeling process and correctly measure the modelers' performance. Nevertheless, existing methods of performance monitoring in building design practices lack an objective measurement system to quantify modeling progress. The widespread utilization of BIM tools presents a unique opportunity to retrieve granular design process data and conduct accurate performance measurements. This research improves upon previous efforts by presenting a novel application programming interface (API)-enabled approach to automatically collect detailed design development data directly from BIM software packages and efficiently calculate several modeling performance measures.

The primary objective of this research is to create and examine the feasibility of a proposed automated design performance monitoring framework. The proposed framework provides the following capabilities: (a) non-intrusive and cost-effective data acquisition for capturing design development events in real time, (b) scalable and high-speed ingestion for the storage of design modeling data, (c) objective measurement of designer performance

and estimating levels of effort required to complete design tasks, and (d) identifying optimal design teams using empirical performance information.

In chapter 3, the utilization of modeling development information embedded in design log files that are produced by Autodesk Revit is proposed as a rich source of performance data. To this end, generalized suffix tree (GST) data structures are utilized to find common, frequent command sequences among Revit users. In addition to identifying the common command execution patterns, the average time it takes the selected modelers to execute command sequences is calculated. The obtained results demonstrate that there is a statistically significant difference between the modelers in terms of the time it takes them to conduct similar modeling tasks.

Chapter 4 utilizes modeling software solution's APIs to automatically collect and store timestamped design development information. The proposed passive data recording approach allows for the real-time capture of comprehensive user interface (UI) interaction and model element modification events. The proposed framework is also implemented as an Autodesk Revit plugin. An experiment is then conducted to verify the accuracy of this plugin. Throughout this experiment, manual recordings of model development events were compared against the automatically generated plugin output.

Chapter 5 outlines the details of an approach to identify the optimal design modeling team configuration based on automatically collected performance data. To this end, an experiment is conducted to capture data using the developed Revit plugin. Experiment participants' individual production rates are estimated to establish the validity of the proposed approach to identify the optimal design team configurations. The presented approach uses the earliest due date (EDD) sequencing rule in combination with the critical

path method (CPM) to calculate the maximum lateness for different design team arrangements.

The primary contributions of this study to the state of knowledge are as follows: (a) proposing a tailored string mining algorithm that is capable of extracting meaningful information from timestamped design development data, (b) developing a framework based on APIs to automatically collect design modeling data, and (c) creating a mathematical model to estimate design modeling project completion times based on individual performance data and project requirements.

This study contributes to the state of practice by (a) allowing design project managers to gain an unprecedented insight into the evolution of a building model using the information embedded in design log files, (b) helping design managers to acquire progress information without the need to manually record and report data, and (c) enabling design managers to identify an optimal modeling team arrangement based on automatically captured, quantitative performance information.

CHAPTER 1: INTRODUCTION

The goal of design is to specify a product that best satisfies the client, ensures safe construction and operations, and achieves minimum overall costs (Wang & Tsai 2011). As capital projects are becoming more complex, the design process increasingly requires substantial interactions among a wide range of designers from various architectural, engineering, and construction (AEC) disciplines (Evins 2013). Throughout this evolutionary process, multidisciplinary teams of architects and engineers need to make difficult decisions to design buildings that are functional, safe, and reliable, and that meet clients' expectations (Anumba & Yang 2013). Given the specific requirements of different disciplines, each specialist has a unique approach to design. The existing variations in the understanding of design problems result in conflicts that negatively impact the concurrent design efforts as well as the downstream construction activities (Simpeh et al. 2015). While design processes account for approximately 5%-10% of the total cost of a typical capital project (Tizani 2011; Egan 1998), rectifying conflicts that result from faulty design decisions accounts for an additional 5%-8% of total project costs (Lee & Pena-Mora 2007). Given the value of the U.S. construction industry (U.S. Census Bureau 2015), approximately $70 billion will be spent annually to resolve design-related issues in capital projects alone.

The poor management of design processes is the primary cause of costly errors in construction projects (Sun & Meng 2009; Love & Li 2000). Similar to other design activities, it is necessary for AEC companies to have an effective performance monitoring system in order to produce accurate design models (Pilehchian et al. 2015). In fact,

effective design progress monitoring is instrumental in preventing potential errors, and it results in both lower overall project costs and productivity improvements across the industry (Riley et al. 2005; CII 2001; CII 2004 McGeorge 1998). Any performance monitoring system depends on metrics to determine the performance of project participants (Figure 1-1). Calculating performance metrics enables managers to identify where team members are falling short, make corrective adjustments, and track outcomes across different projects (Chiu & Russel 2011; Skibniewski & Ghosh 2009).

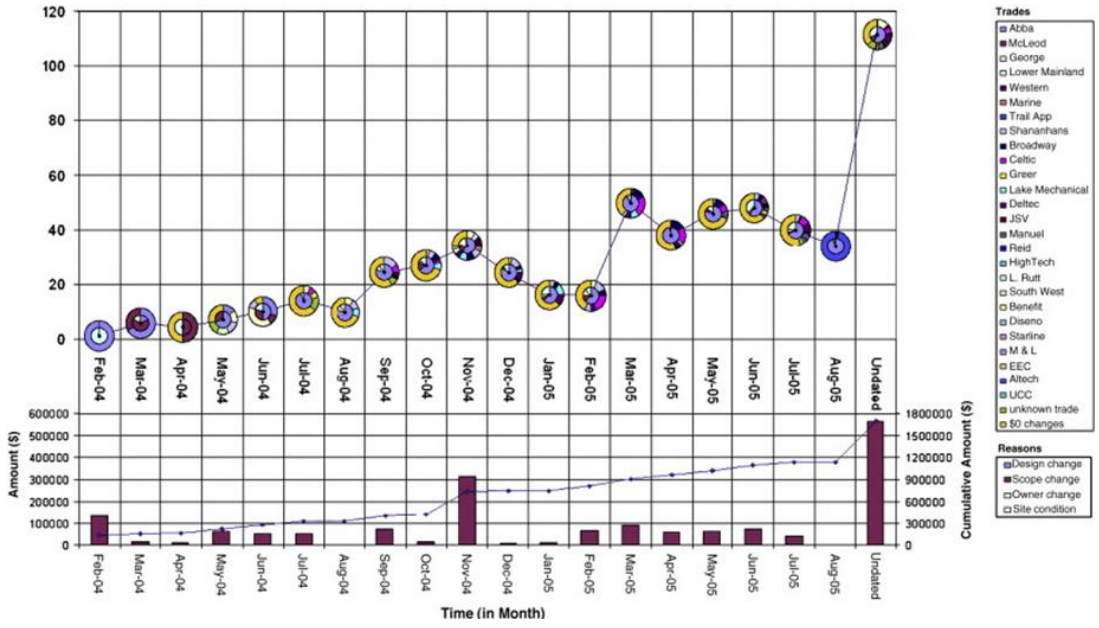

Figure 1-1 Design Performance Monitoring Dashboard (Chiu & Russel, 2011)

1.1 Current Performance Monitoring Methods for Building Design Projects

The existing literature on performance monitoring in design practices can be categorized as indicated in Table 1-1. Design progress is traditionally measured by tracking a designer's or engineer's production rate as the relationship between physical inputs and outputs. The current design or engineering performance metrics can be classified as follows (Ashuri et al. 2014):

- *Design Hours per Construction Document*—in this approach, design hours and construction documents (for example, drawings, specifications, and contract forms) are considered to be process input and output respectively. In a study to measure engineering productivity, Thomas et al. (1999) proposed using design work hours per drawing sheet, design work hours per specification section, and design work hours per contract document to measure design progress. Chang and Ibbs (2006) measured production rate using design work hours per drawing sheet to identify the major factors that affect design productivity. These metrics regard project documents as tangible outputs of design, which makes output estimation relatively less burdensome. The number of billable hours that designers spend can also be measured using company payroll information. However, this method does not account for the differing complexity and unique characteristics of construction projects. Therefore, the proposed metrics are better suited for comparing design performance across similar projects.

- *Design Hours per Installed/Build Quantities*—this approach considers design hours and installed or built quantities (for example, the amount of installed equipment, concrete volume, and building floor area) as the input and output of design processes respectively. In an effort to measure performance in 10 engineering disciplines, the Construction Industry Institute (CII) (2001) proposed several trade-specific metrics based on the number of equipment pieces designed. Kim (2007) later used these metrics in another CII-supported project to benchmark engineering performance. Sacks and Barack (2008) and Sacks et al. (2010) are other examples of utilizing installed or build quantities to estimate design output, where the authors investigated the impact of 3D parametric modeling on structural engineering productivity. The methodologies proposed in these studies incorporate project characteristics and design quality in the evaluation of design performance; therefore, they can be used to compare projects that are different in nature.

- *Normalized Design Hours*—in this approach, design hours are normalized using a basis for design hours. The CII (2004) developed multiple regression models to calculate the basic hours in different engineering disciplines, and design performance in each discipline is measured by normalizing the actual design hours against the calculated basis hours. A normalized metric of less than one indicates a performance that is better than the benchmark. Liao et al. (2009) and Liao et al. (2011) proposed modifications to the CII's methodology to convert and aggregate unit-less design metrics into project-, portfolio-, and company-

level measures. This allows managers to compare performance not only across different projects but also across different disciplines.

Table 1-1 Overview of Existing Literature

Article	Metric	Required Data
Thomas et al. 1999 *Chang and Ibbs 2006*	Design hours per construction document	Design hours Number of design and construction documents (for example, drawings, specifications, and contract forms) for different disciplines
CII 2001 *Kim 2007* *Sacks and Barak 2008* *Li et al. 2014*	Design hours per installed or built quantities	Design hours Installed or built quantities (for example, linear feet of pipe, concrete volume, and building floor area) for different disciplines
CII 2004 *Liao et al. 2009* *Liao et al. 2011*	Normalized design hours	Design hours per installed or built quantities

While the above studies primarily rely on input and output production metrics, some researchers utilized a more holistic perspective and included other measures, such as design quality, design innovation, design coordination, and client satisfaction, in evaluating design projects (Yarmohammadi & Ashuri 2015; Ren et al. 2013; Torbett et al. 2001; Tucker & Scarlett 1986). For example, metrics, such as delay and cost overrun in fabrication and construction due to design deficiencies or constructability issues, were utilized to evaluate the outcomes of design projects (Lu et al. 2014; Bassioni et al. 2005; Fayek & Sun 2001).

Overall, the existing methods for design progress monitoring are not up to date, i.e., there is a significant lag between the time at which these progress metrics can be calculated and the time at which design decisions need to be made throughout various phases of design development. These methods only consider what has been spent and produced in design projects without sufficient appreciation for the complexity of design evolution as an evolving system. Several studies have noted this limitation and indicated that the AEC industry needs to adopt a forward-looking approach to enable proactive design monitoring (Du et al. 2014; Succar et al. 2012; Leong & Tilley 2008). Due to this major drawback, current progress monitoring methods have limited capability for helping design managers to monitor design development in real time and detect design deficiencies in order to take timely corrective actions.

Above all, the greatest challenge in monitoring design processes is the lack of an objective and systematic method to accurately capture the required data for quantifying progress in design modeling. Several studies have indicated that existing data collection approaches are time consuming, manual, and incapable of capturing information in real time (Knotten & Svalestuen 2014; Park et al. 2013; Navon & Sacks 2007); therefore, they are inherently incapable of retrieving useful information, created at the level of various design tasks, such as conceptual layouts, model element design, and model detailing (Kymmell 2008; Meredith & Mante 2003). Moreover, while the widespread application of computer-aided design in the full range of design activities (for example, conceptual design, detailed design, and construction documents) has facilitated knowledge integration from various participants, collecting design modeling data remains a challenging task (Volk et al. 2014).

1.2 Performance Monitoring in Collaborative Computer-Aided Design

Collaboration in design is essential for project success; however, the circulation of incomplete and erroneous process information among project stakeholders makes design issues inevitable (Love et al. 2014; Anumba et al. 2002; Abdul-Rahman 1995). In addition to challenges related to collaboration, the following are among the common causes of design problems: the completeness of scope definition, the project objectives and priorities, owner profiles, and the reliability of vendor data (Doloi 2012; Chalabi et al. 1987). The availability of comprehensive information from all disciplines as well as learning from past projects are necessary for project managers to anticipate problems in design development and to take corrective actions in a timely manner (White et al. 2005).

Early research in collaborative design management was qualitative in nature, and it was often based on studying a single design case. Earlier works attempted to address design issues in the context of a recommendation for a given product. In these studies, there have been frequent references to the idea that by making use of recommended techniques, one can improve the productivity, quality, and performance of designers or engineers; for examples, see concurrent engineering practices (Benayoune & McGreavy 1994; Isbell 1993) and civil engineering design (Girczyc & Carlson 1993; Winter 1992; Graham 1990; Sackett & Evans 1984). Even though these studies offer several recommendations for improving design performance, they fail to acknowledge the inherent variations in approaches that designers in a particular field or from different disciplines take to tackle a design problem. The unique nature of each design and construction project makes it difficult to address specific design problems by implementing generic suggestions (Alzraiee et al. 2012). A single set of rigid design strategies could not be applicable to the

construction industry, in contrast to the manufacturing industry, which deals with highly repetitive processes and mass production, i.e., 43% productive time in construction vs. 88% productive time in manufacturing (McGraw-Hill 2013).

Computer-aided design tools provide practitioners with an information technology enabled approach that involves applying and maintaining an integral digital representation of all building information for different phases of the project lifecycle in the form of a data repository (Gu & London 2010). The capabilities of virtual design technologies have eased collaboration in typical design teams that involve a wide array of disciplines, such as architecture and structural, seismic, hydraulic, and pipeline engineering, working together for a relatively short period of time (Plume & Mitchell 2007). To facilitate virtual design efforts, different sequential and parallel collaboration strategies have been proposed for implementation (ENR 2013; Eastman et al. 2011; Korman et al. 2008). A number of studies have been conducted to measure the success of these strategies to improve performance. Lee and Kim (2014) conducted a case study of a seven-story office building to investigate the impact of parallel versus sequential approaches on a design coordination team's production rate. Their findings indicated that a sequential design strategy is faster than the parallel strategy in terms of design productivity. A further examination of these two approaches identified deficient information sharing among design team members as the main factor that negatively impacts performance. Other case studies, such as those conducted by Staub-French and Khanzode (2007) and Manning and Messner (2008), also investigated collaborative strategies with a focus on evaluating the impact of virtual design and construction (VDC) solutions on design performance. Together, these studies

demonstrated that information transfer bottlenecks are the primary challenges of design progress monitoring in collaborative computer-aided design.

An important observation is that the data accumulated in building models largely consist of information about different building systems (for example, structural, mechanical, electrical, and architectural), and they exclude model progress data. In fact, AEC companies still use conventional manual practices, such meeting minutes reports and Gantt charts, to document design modeling progress data (Yarmohammadi & Ashuri 2015). In larger projects, those reports consist of several pages of emails, charts, descriptions, and spreadsheets that are difficult to read and analyze (Dave et al. 2016). These unstructured documents lack the organization necessary for machine readability—i.e., inclusion in a relational database that search engine algorithms can readily search (Baars & Kemper 2008). Structured data are understandable in machine language, and computers can automatically read and analyze them. In contrast, unstructured data are only understandable to humans, who do not interact with information in strict, database formats (Gautam & Yadav 2014). The manual compilation of unstructured progress reports to measure design modeling performance is a time- and energy-consuming task.

Due to the difficulties associated with progress measurement in design modeling, most decision makers rely heavily on subjective measures and informal communications to assess progress (Bate & Robert 2007). For instance, when a structural designer reports to the design manager that 30% of the steel framing has been modeled, the determination of the percentage of completion is primarily based on the designer's experience, and it does not present the real progress of the project in an objective manner. The resulting human error in the preparation of progress data and their delivery to design managers causes

inefficiency in performance monitoring (Taylor-Adams & Kirwan 2013). Moreover, inaccurate measurements that are calculated using faulty data can be misleading to managers, thereby causing them to make decisions with negative impacts on the design modeling process.

1.3 Chapter Summary

As the complexity of capital projects grows, the design modeling process increasingly involves massive collaborative efforts among various AEC disciplines. Throughout this multidisciplinary process, the various priorities of design team members often result in conflicts that can negatively impact project outcomes. To prevent such conflicts, managers need to closely monitor the design modeling process. However, a review of the existing literature indicates that the current methods of design performance monitoring lack objective measurement systems to quantify modeling progress. The difficulties associated with evaluating design modeling performance renders the existing methodologies impractical.

CHAPTER 2: PROBLEM STATEMENT AND RESEARCH OBJECTIVES

2.1 Motivation and Gaps in Knowledge

Overall, the existing literature highlights the crucial role of performance management in the success of building design processes. By utilizing the metrics proposed in the existing studies, design managers can monitor their teams by comparing their performance against established baselines. However, the discussed studies make no attempt to address the issue of data collection during the design modeling processes. The gaps in the current state of knowledge that render most of the existing approaches impractical are as follows: the existing design monitoring practices have delays, these practices are manual and labor-intensive, the manually collected design monitoring data are of a low quality, and the existing methods are subjective and judgment-based. These four gaps are discussed next.

2.1.1 Gap 1 in Knowledge: Existing Design Monitoring Practices Have Delays

The existing design modeling performance approaches are either backward-focused or trailing. There is a significant lag between the time at which the measures are calculated and the time at which design modeling happens (Jansson et al. 2016). As illustrated in Figure 2-1, there is also a delay between the time when progress data are captured and when they are reported to the management. Such delays in current building modeling practices prevent design managers from taking corrective actions in a timely manner

(Yarmohammadi et al. 2017) because it might be too late or too costly to make any corrections by the time the team reacts to these lagging indicators.

There is a need for a data acquisition method that can provide design managers with access to real-time, building model development information. Such a system should be able to capture large amounts of accurate model progress data from project participants across different stages of design. Once analyzed, the captured granular information can provide managers with a comprehensive view of the state of their project compared to the predetermined milestones. Additionally, leading performance indicators can be measured using real-time design progress data.

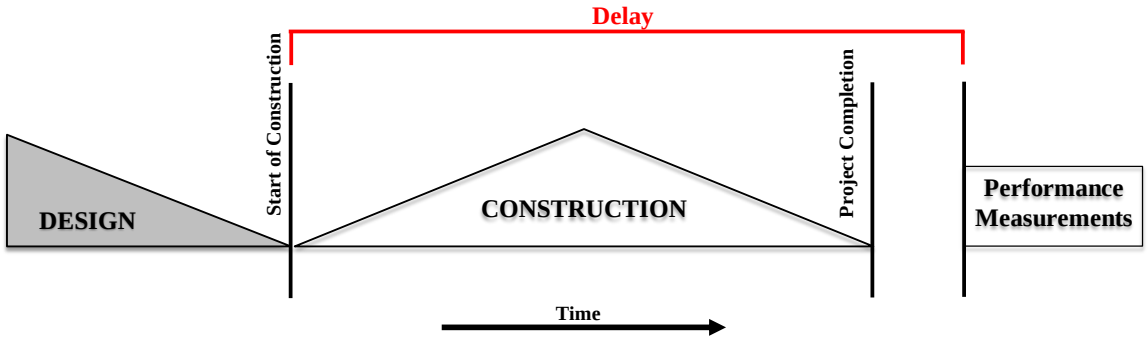

Figure 2-1 Lagging Indicators in Existing Design Modeling Monitoring Practices

The lagging measurements of design performance metrics can provide information about a project after the fact. However, the question remains regarding the value of these metrics as future predictors for design-related conflicts (Hinze et al. 2013). Moreover, an unbalanced focus on lagging after-the-fact-based approaches may discourage conflict prevention (Jin et al. 2013). Therefore, organizations have also adopted proactive leading

measures to predict future levels of performance. For such systems to function properly, it is necessary to collect accurate data in real time (Manyika et al. 2011).

2.1.2 Gap 2 in Knowledge: Existing Design Monitoring Practices Are Manual and Labor-Intensive

Existing methods require extensive manual data extraction from various data sources, such as design documents, schedule and budget updates, and status and progress emails. In practice, most design managers informally collect information about the state of the project from the members of the design team at different time intervals (De Marco & Narbaev 2013).

Table 2-1 presents a summary of the metrics and corresponding information items used in the existing literature to measure progress in design practices. An important observation is that the collected information items vary based on the metrics used to calculate design performance. The required information is usually manually extracted from various sources to obtain the necessary quantities.

Manual data collection is slow and inaccurate, and it produces vast amounts of paperwork (Yarmohammadi et al. 2017). In addition to difficulties associated with manually collecting the required data, design team members generally lack the incentive to record and report information. This is largely because designers are expected to complete the job under tight schedules and are hence left with little time to keep track of performance data (Ding et al. 2014). This task becomes even more challenging, since the disparities among numerous project participants make it difficult for design managers to efficiently capture the necessary information (Liao et al. 2011).

2.1.3 Gap 3 in Knowledge: Manually Collected Design Monitoring Data Are of a Low Quality

The manual nature of current monitoring methods may undermine the quality of the measurements because they introduce error and bias into the process. The design manager needs to compile large amounts of data from multiple sources and provided by various parties involved in the design process (Yin et al. 2011; Sacks & Barak 2007). The excessive amount of work required to extract information from the provided data sources and perform the necessary analyses may cause human errors that reduce the quality of the resulting measures. The collected information also presents a designer's interpretation of what needs to be measured and the way in which this measurement must be conducted; therefore, it may not reveal the actual state of the project. Such drawbacks, as well as difficulties associated with capturing the information required for accurate measurement, have rendered existing design monitoring methods impractical (Yang et al. 2010). The design manager consequently relies heavily on experience and informal communications with the design team to monitor design projects (Bate & Robert 2007).

2.1.4 Gap 4 in Knowledge: Existing Design Monitoring Methods Are Subjective and Judgment-Based

An accurate measurement of the design progress represents one of the most challenging data gathering problems when monitoring design projects (White et al. 2005). It was reported that there is a tendency among members of the design team to use as-planned progress goals as proxy measures for actual progress or to select only those metrics that allow for favorable progress to be reported (Meredith & Mantel 2003). This problem

is more widespread when non-systematic monitoring methods, such as weighted milestones and budget-based monitoring, are used to track design progress. For instance, a structural system designer reports to the design manager that 30% of the steel framing has been modeled. In this case, the following questions are not clear with regard to the measure that has been utilized to evaluate progress: does that figure imply that 30% of the planned design hours have been spent, is it 30% of the required design documents that have been prepared, or does it mean that 30% of the level of development (LoD) specified in the contract has been achieved?

In addition, the determination of the percentage of completion is primarily based on the experience of the designers; therefore, it does not present the real progress of the project in an objective manner (Sacks & Barak 2008). Subjective progress reports may also be biased as different design disciplines may not reveal the truth considering incentives, penalties, and other project-specific conditions. As a result, subjective monitoring methods can be misinterpreted, and they can mislead the design project manager, since discrepancies between the as-planned and actual design progress remain undetected and could lead to further conflicts throughout the project. Also, relying on subjective measures to determine design progress is the main source of several problems in planning design projects, such as setting unreasonable expectations for design completion and misallocating resources to the project (Shahtaheri et al. 2014). For instance, in a case study, Leite et al. (2011) challenged the conventional assumption about the effort required to generate a design model in a required LoD, and they demonstrated that more detail in the model does not necessarily mean a proportionally higher modeling effort.

Table 2-1 List of Metrics and Information Items Used in The Existing Literature

		Productivity Metrics					
		Thomas et al. (1999)	CII (2001)	CII (2003)	Chang and Ibbs (2006)	Kim (2007)	Sacks and Barak (2008, 2010)
		design hours per sheet, design hours per section, design hours per each contract document	direct work hours* per installed unit	Direct design hours*/basis hours** predicted	design hours per drawing	direct work hours*/Issued For Construction (IFC) quantities	design hours per 1000 m^2 of floor area, design hours per m^3 of precast concrete
Information Items	Design Hours	✔	✔	✔	✔	✔	✔
	Number of Drawings	✔			✔		✔
	Number of Contractual Documents	✔					
	Project Size	✔	✔	✔	✔	✔	✔
	Project Type	✔	✔	✔	✔	✔	✔
	Project Cost				✔		
	Client/Owner		✔	✔		✔	
	Number of Installed Pieces		✔	✔		✔	✔
	Volume of Structural Concrete			✔		✔	✔
	Project Delivery System		✔	✔		✔	
	Project Profit				✔		
	QA/QC				✔		

2.2 Research Objectives

The emergence of open-access design software packages provides a unique opportunity to extend the core capabilities of VDC modeling tools. In particular, these software solutions can be utilized to record design development events and UI interactions with design models. However, no research has been conducted to use such capabilities for monitoring design progress. In simple words, various members of the design team leave traces behind as they interact with virtual design models. Historical records of design development events (for example, changes in design elements, and executed commands by each user) provide a rich source of information about the progress of a design project.

This dissertation seeks to investigate the possibility of utilizing the data of user interactions, as well as design development events, to extract and measure useful information regarding the design modeling process. Such information will be used to examine the research hypothesis that there are meaningful differences among modelers in terms of the time it takes them to conduct similar modeling tasks. Additionally, the extracted information can be utilized to measure the design modeling progress at the project level and to benchmark the performance of each team member involved in the development of the design model.

The motivation behind the proposed research is that design software protocols, such as APIs, can be devised to create a novel data acquisition method. Once the system is placed in the design environment, a wide range of design development events (for example, attributes related the performed design tasks, and UI interaction features) are captured in real time and automatically transferred to a database of information that is required for design monitoring.

This research is the first attempt to introduce the use of design software APIs as an alternative source of information for monitoring design modeling progress and analyzing modelers' performance. The fundamentally different property of the proposed approach is its adaptability to the development of design models that, by nature, are complex, dynamic, and evolving. This overall research objective is broken down into three sub-research objectives as follows: (I) to examine the feasibility of extracting modeler performance information from design log files, (II) to examine the feasibility of automatically extracting modeler performance information using APIs, and (III) to identify optimal design modeling teams using performance information. These objectives are discussed in detail below.

2.2.1 Research Objective I: To Examine the Feasibility of Extracting Modeler Performance Information from Design Log Files

Due to the difficulties associated with the manual collection of design modeling progress data, the first research objective of this study is to empirically examine the feasibility of utilizing modeling log data to collect granular performance information. The specific tasks to accomplish this objective are as follows: (1) to investigate the presence of frequent command sequences (i.e., patterns) that represent specific modeling tasks in design log files, and (2) to empirically characterize the performance of modelers based on the extracted information. The non-intrusive and cost-effective data acquisition capability of the proposed approach will be used to identify and characterize performance variations observed among design modelers. The following questions are of interest:

- What sequences (patterns) are formed from various commands?
- What types of command sequences do designers execute frequently?

- What patterns are common among different designers?
- Is there a meaningful difference between the average time it takes designers to execute different command sequences?

The results of this part of the study are expected to enable design project managers to empirically evaluate, benchmark, and compare the performance of their modelers across different projects.

2.2.2 Research Objective II: To Examine the Feasibility of Automatically Extracting Modeler Performance Information Using APIs

The second objective of this study is to devise and implement a methodology to automatically extract modeler performance information utilizing software application program interfaces. API-based information extraction can help design management teams to directly collect granular data from design software solutions in real time. The information acquired using this methodology includes changes in design elements, executed commands by each user, and errors.

2.2.3 Research Objective III: To Identify Optimal Design Modeling Teams Using Performance Information

The final objective of this study is to develop a data-driven approach to identify optimum modeling team configuration based on performance information captured through a software API. The mathematical model developed in this section helps project managers to choose a modeling team that is the best fit for the project at hand.

2.3 Research Methodology

This dissertation aims to address the three presented research objectives in three separate sections. The research methodology for each research objective is briefly presented in this section, and full details are provided in the corresponding chapters.

To address the first research objective, a sequential pattern mining (SPM)-based approach to retrieve command execution patterns is developed. The following steps are taken to develop the proposed approach:

- Collect design log files from a major international architectural design firm.
- Extract the required performance information from log files using a tailored text parser.
- Convert the collected data to a form appropriate to be used in string mining algorithms.
- Implement a GST-based pattern mining algorithm to identify common command patterns.
- Calculate the average times it takes modelers to execute identified patterns to evaluate the existing performance levels.
- Examine the possibility of using design log files as rich sources of modeling performance data by conducting statistical tests.

The second research objective is addressed by utilizing design software solution APIs to automatically collect and store timestamped design development information. The research methodology of this chapter consists of the following steps:

- Identify an API functionality capable of recording information in real time without disrupting design modeling processes.
- Design a modeling data collection framework by utilizing the event-handling functionality of software APIs.
- Develop a Revit plugin that is capable of capturing and storing UI interactions and model element modification events in real time.
- Verify the functionality of the developed plugin.

The plugin developed in this chapter should capture all possible software actions, ranging from creating, selecting, and modifying elements to navigating through different zones of a virtual model. Such recordings, constituting timestamped event sequences, are organized and stored in searchable databases.

To address the final research objective, a mathematical model is utilized to choose and assign modelers to projects based on their past performance information. The following steps outline this chapter's research methodology:

- Design and conduct an experiment to collect design modeling data utilizing the plugin developed in the previous chapter.
- Analyze the collected data to evaluate individual production rates.
- Identify the optimal modeling team configuration to minimize lateness utilizing the EDD approach.

2.4 Dissertation Organization

To achieve the above-mentioned research objectives, the remainder of this dissertation is structured as illustrated in Figure 2-2. Chapter 3 addresses the first objective,

in which software-generated design log files are analyzed to identify command execution patterns and measure task execution times to evaluate modelers' performance levels.

The second research objective of this dissertation is addressed in chapter 4. A framework to directly capture performance data from software solutions is proposed. The proposed approach, which uses event handlers to record the evolution of design models, is implemented in a .NET integrated development environment.

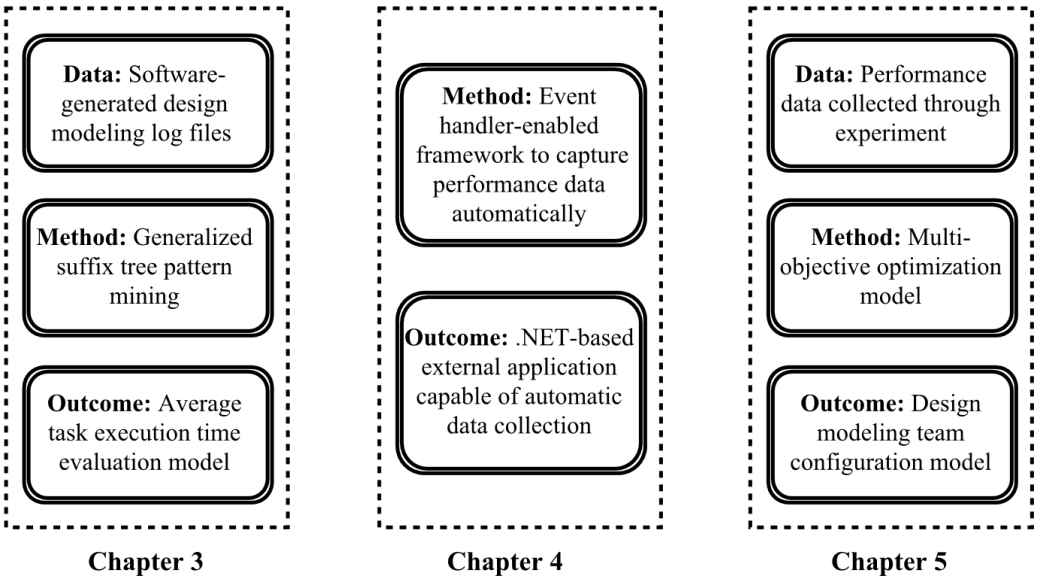

Figure 2-2 Chapter Contents

A methodology to identify the optimal design modeling team is presented in chapter 5. The external application, developed in the previous section, is used to collect performance data and calculate individual production rates. A mathematical model identifies the configuration of the optimal design modeling team based on captured performance information. Finally, chapter 6 concludes the research works presented in this dissertation, and possible future works and extensions to the proposed analysis are suggested.

2.5 Chapter Summary

The existing literature highlights the important role of performance management in the success of design modeling processes. However, current methodologies do not address the problems associated with collecting design modeling data that make evaluating performance metrics difficult. These problems can be summarized as follows:

- There is a significant delay between design modeling data collection and performance evaluation.
- Collecting design modeling performance data is a manual and labor-intensive process.
- Manually collected design modeling performance data are of a low quality.
- The existing design performance management practices rely mainly on subjective and judgement-based information.

This dissertation seeks to investigate the possibility of utilizing the data of user interactions, as well as design development events, to extract and measure useful information regarding the design modeling process. Such information will be used to examine the research hypothesis that there are meaningful differences among modelers in terms of the time it takes them to conduct similar modeling tasks. The specific research objectives of this dissertation are as follows:

I. Examine the feasibility of extracting modeler performance information from design log files.

II. Examine the feasibility of automatically extracting modeler performance information using APIs.

III. Identify optimal design modeling teams using performance information.

The first research objective is addressed in chapter 3, where software-generated modeling log files are analyzed to identify command execution patterns and measure task execution times to evaluate modelers' performance levels. Chapter 4 addresses the second objective; in this chapter, a framework to directly retrieve and store performance data from modeling software tools is proposed. The proposed approach utilizes event-handling functionality to record the evolution of design models. In chapter 5, an approach to identifying the optimal design modeling team is presented. This objective is achieved by optimizing the design team configuration using a combination of scheduling and sequencing methodologies. A summary of research findings and some recommendations on future research directions are presented in the final chapter.

CHAPTER 3: EXTRACTING MODELER PERFORMANCE INFORMATION FROM DESIGN LOG FILES

3.1 Introduction

The ability to collect objective progress data is critical to accurately measure performance in design modeling practices. This chapter presents an SPM-based approach to retrieving data embedded in design log files. The research objectives of this section are to (1) investigate the presence of frequent command sequences (i.e., patterns) that represent specific modeling tasks in design log files, and (2) empirically characterize the performance of modelers based on the extracted information.

Throughout this chapter, several steps that are necessary to retrieve performance information from design log files, including data collection, data preparation, and frequent command pattern extraction, are explained. The Atlanta office of a major international architectural design firm provided the data for this chapter. Over 11 million user-model interaction records were analyzed to find common command execution patterns among building modelers. The novel method created in this chapter contributes to the body of knowledge by incorporating the chronological dependencies of textual records into the existing pattern matching models. The findings presented in this chapter contribute to the state of practice by enabling design project managers to empirically evaluate and compare the production rate of their modeling team members.

The rest of this chapter is organized as into three broad components. The research methodology used to find modeling patterns in design log files is explained next. Then, a discussion of the results is provided, followed by a conclusion in the final part.

3.2 Mining Design Log Files

The widespread utilization of VDC tools presents AEC practitioners with an unprecedented opportunity to automatically capture objective, fine-grained performance data (Boton et al 2015). Design log files generated by software packages such as Autodesk Revit and Tekla Structures can be rich sources of process-specific information. These files are unstructured text files that capture all activities that occur during a modeling session (Autodesk 2016). Mining the information stored in these files can provide practitioners with a unique insight into the evolution of a building model (Yarmohammadi et al. 2016). However, existing applications of data mining methods in the context of the AEC industry do not incorporate timestamps of unstructured textual data into their analyses.

Soibelman and Kim (2002) outlined the steps necessary to apply data mining and knowledge discovery in databases (KDD) as tools to extract novel patterns in the design and construction fields. In two consecutive studies, Caldas et al. (2002) and Caldas and Soibelman (2003) proposed text mining-based approaches to automatically classify unstructured construction documents. These efforts laid a solid foundation for a methodology that Caldas et al. (2005) developed to retrieve project documents (for example, requests for information [RFIs], change orders, and design reviews) based on the building model element to which they correspond. Fan et al. (2014) proposed an extended

information retrieval system capable of classifying unstructured documents based on their corresponding projects. Construction cost overrun prediction by William and Gong (2014) is another example of implementing data mining algorithms to analyze unstructured textual data. They used a stacking ensemble model of several classifiers to forecast the level of cost overruns. While these studies offer valuable insights into how to overcome several challenges in handling unstructured textual AEC data, they do not address the way in which the chronological dependencies of temporal data can be incorporated into the analysis. This is a significant limitation as the temporal aspect of unstructured textual data stored in design log files captures the progress of design modeling. Due to this shortcoming, the discussed methods cannot be utilized to mine meaningful information from design log files, whereas sequential pattern mining methodologies have the capability to analyze and retrieve information from timestamped textual data.

3.3 Research Methodology

3.3.1 Dataset

The presented study utilized design log data from an international architectural design firm with operational expertise in multidisciplinary practices, including architecture, interiors, urban design, and landscape architecture. Journal files generated by Autodesk Revit software were collected to conduct the analysis. These files capture all modeling activities that occur during a design session as well as system information, such as memory and processor usage. Revit journal files are largely used to diagnose and troubleshoot technical problems. However, in this research, these log files were utilized as a non-intrusive, data capturing mechanism for documenting designer-software interactions

and recording model development events. The author had access to data from healthcare projects that were designed in 2013 and 2014. The provided database consisted of over 4,000 Revit journal files that were later parsed to extract and store the recordings of executed commands, amounting to over 10 GB of structured data. In addition to the modeling events, these journal files contained information on the modeler's identity and the projects that the models belonged to.

The information extracted from the Revit journal files was utilized to examine the research hypothesis that there are meaningful differences among modelers in terms of the time it takes them to conduct similar modeling tasks. In the context of this research, a journal log file is regarded as a database of ordered modeling events (commands), recorded with a concrete notion of time; a frequent pattern is an ordered set of individual commands that occur more than a threshold number of times (i.e., minimum support) in the original sequence database; and minimum support is an indication of how frequently a pattern appears in the database.

The following questions are of interest in this research:

- What types of commands sequences do modelers execute frequently?
- What structures are formed from various commands at each stage of modeling, and how?
- What are the command pattern sequences common among different modelers?
- What types of modeling tasks do these patterns represent?
- Is there a detectable difference between the time it takes BIM users to execute common pattern sequences?

3.3.2 Algorithm

Log files (or transaction logs) have long been studied in the data mining community. These files can be generated in different applications, such as retail transactions data and web access logs. Srikant and Agrawal (1996) introduced the generalized sequential pattern (GSP) algorithm to mine shopping patterns. The authors studied a large database of customer transaction data, where each transaction consisted of a customer-ID, a transaction time, and the items bought in the transactions. Compared to exhaustive search methods, the GSP significantly reduces the search space by utilizing a downward closure property, which guarantees that for a frequent set of items, all its subsets are also frequent, and hence infrequent sets can be removed from the search space without affecting the results (Pei et al. 2001a). However, the time and memory performance of the GSP is relatively low as a large number of candidates must be generated and stored in each repetition for evaluation (Verma & Mehta 2014). Other general methodologies to mine frequent subsequences, such as the PrefixSpan algorithm (Pei et al. 2001b) and CloSpan (Yan et al. 2003), further reduce the size of the search space by taking advantage of divide-and-conquer approaches. However, these SPM methodologies do not preserve the exact order of elements in a sequence. Therefore, some of the elements in extracted sequences may not necessarily be consecutive in the original string of transactions. In this chapter, the author used a special SPM algorithm, called a GST, that maintains the order in which the executed commands are recorded.

A major objective of this dissertation is to characterize BIM users based on the time it takes them to execute modeling tasks. Therefore, in addition to investigating different commands and calculating their statistics individually, modelers are characterized based

on the common sequence of commands they execute. For example, by looking at each type of command individually, it is unlikely to determine how quickly a user performs the task of "modifying the position of an object." Instead, this measurement can be obtained by calculating the average time it takes the modeler to perform the "select-rotate-move" command sequence on a model element.

The GST can extract such patterns, since it preserves the original order of recorded transactions. Xiao and Dunham (2001) first proposed applying a GST data structure to the mining of web access log data. The authors analyzed clickstream data, which was generated based on the access by Internet users, to identify frequent web page traversal patterns. The technique proposed by Xiao and Dunham achieved a high level of adaptability to large databases through dynamic compressions and effective pruning. Guerbas et al. (2013) introduced an improved version of the GST algorithm with an optimized data structure to extract the page visit patterns of Internet users. The authors' primary objective was to improve the search experience of users by predicting the pages they intended to view next. To this end, they utilized the GST algorithm to identify common web navigational patterns among users with similar interests. The methodology utilized in this chapter is a modified version of Guerbas et al.'s algorithm that is tailored for mining journal log files.

To better understand how GST data structures are built, consider two strings of characters "abab" and "aab". As presented in Table 3-1, unique identifiers "$" and "#" are added to each string. Each character is assigned an index number starting from 1. Also, all possible suffixes of these two strings, including their unique identifiers, are listed in the table.

Table 3-1 Sample String Suffixes

	String 1	String 2
	abab$	aab#
Character Position Index	12345	1234
All Possible Suffixes (Substrings)	$ b$ ab$ bab$ abab$	# b# ab# aab#

A suffix tree "*T* for *m*-character string *S* is a rooted, directed tree with exactly *m* leaves numbered 1 to *m*" (Gusfield 1997). In this example, string 1 has three unique characters "a", "b", and "$". Therefore, as indicated in Figure 3-1, the root node (N_1) will have three children. To continue building the tree, the longest path from the root, which matches a prefix of each suffix, should be found. Then, the suffix structure is built by adding leaves to the existing node. For instance, "b$" and "bab$" are represented by adding node "N_3" to prefix "b".

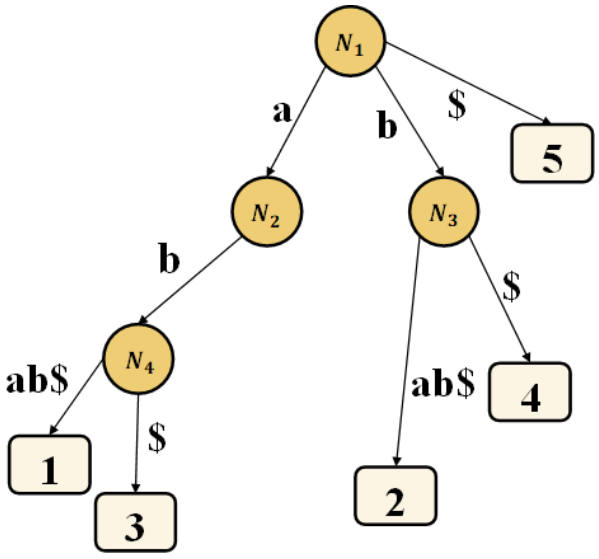

Figure 3-1 Suffix Tree Data Structure for One String

The remaining characters of each suffix (i.e., "$" and "ab$") are then added as children of internal node "N3". The numbers in the boxes at the end of each branch represent the starting character's position index. For instance, suffix "bab$" (represented on path N_1-N_2) starts at index number 2 while suffix "$" starts at the last index.

The same steps should be repeated to add string 2 to the existing suffix tree data structure. String 2 has three unique characters "a", "b", and "#". As illustrated in Figure 3-2, only one more branch needs to be added to the root node "N1," since it already has two children representing "a" and "b". As with string 1, the remaining suffixes are added by searching for the longest path from the root that matches a prefix of the substrings. The starting character indexes are then added to the end of each node-to-leaf path. This step is continued until all suffixes are added to the tree. The resulting data structure is a GST (Figure 3-2).

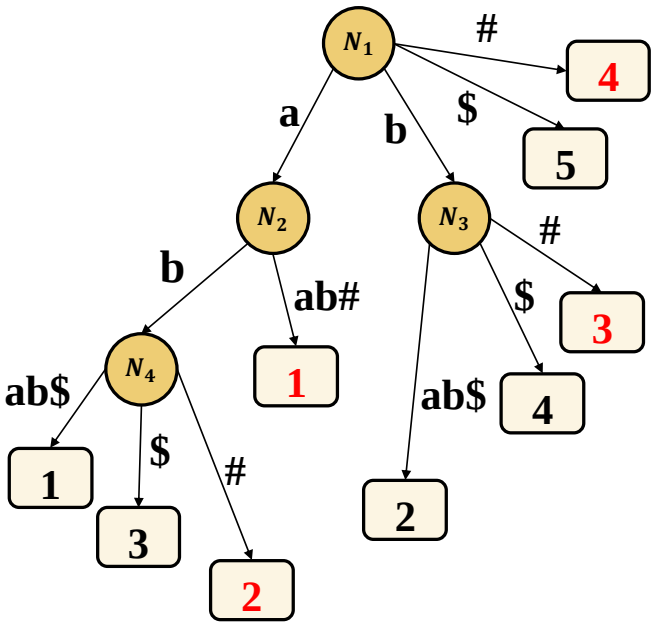

Figure 3-2 A GST Data Structure for Multiple Strings

The constructed data structure can be used to find the number of occurrences of each suffix in multiple strings. For instance, substring “ab” is on the N1-N2-N4 path; therefore, N4’s children should be counted to calculate the total number of times “ab” appears in the strings 1 and 2. In fact, a simple check confirms that “ab” appears a total of two times in the two analyzed strings. The implementation details, as well as the obtained results, are discussed in the following sections.

3.3 Implementation

To fulfill the objectives of this chapter, the required steps to extract the necessary information from Revit journal files and identify common frequent command execution sequences are presented in Figure 3-3. This process consists of three major parts. First, a large number of Revit journal files that belong to a design project were collected and parsed

to extract and store necessary data items. Second, the obtained data were transformed to construct long strings of characters and generate input vectors. Finally, the GST data structures for each user were constructed to retrieve frequent patterns and estimate task execution times.

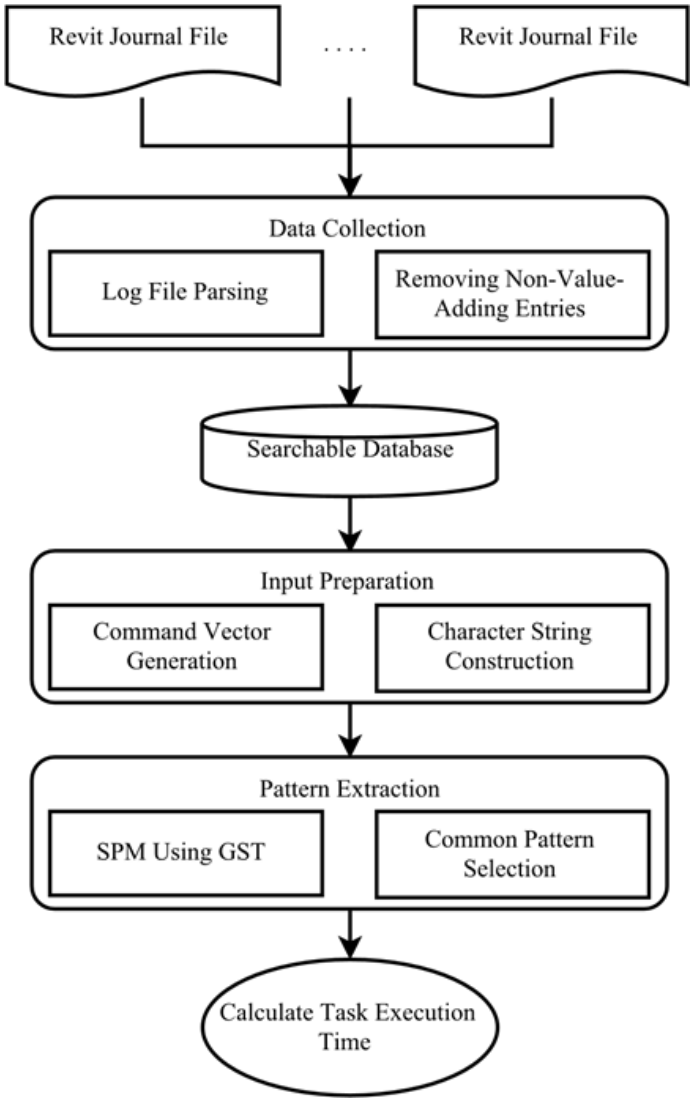

Figure 3-3 Proposed Methodology for Extracting Information from Revit Journal Files and Identifying Common Frequent Command Execution Sequences

3.3.1 Data Collection

The temporal command execution data items were extracted, including the user ID, date, time, project name, Revit version, view type, and command description, from several journal files that the industry partner provided. This step was particularly challenging as there is no documentation available on a public domain that specifies how and where different data components are recorded. Table 3-2 provides examples of each data item as it appears in the journal files. The files were manually searched to identify the local format by which each data instance (for example, project name, command, or view type) is recorded.

Once these protocols were identified, a text processor was developed in the Python programming language. The developed text parser uses regular expressions to extract and store information in a comma-separated values (CSV) file, as presented in Table 3-3. Non-value-adding commands, such as "cancel the current operation" or "delete", were removed from the stored entries to improve the quality of the obtained command patterns.

Table 3-2 Examples of Journal File Recorded Data

Command Name	Example
User Name	`Jrn.Directive "Username" _` `, "Modeler1"`
Date and Time	`'E 29-Mar-2014 10:26:05.808;  0:<` `Jrn.Command "Internal" , "Show or hide recent files , ID_STARTUP_PAGE"`
Project Name	`Jrn.Data "File Name" _` `, "IDOK", "P:\Atlanta\gatech\building.rvt"`
General Command Description	`Jrn.Command "AccelKey" , "Cut the selection and put it on the Clipboard` `' 0:< <<Begin build CT>>`
Specific Command Description	`:13.619;  0:<` `y" , "Cut the selection and put it on the Clipboard , ID_EDIT_CUT"`
View	`ective "WindowSize" _` `, "[2013_03_21_NGHA_TAIF_blochaa.rvt]", "Floor Plan: LEVEL 01 PLANNING"`
Software Version	`Jrn.Directive "Version" _` `, "2014.000", "2.122"`

Table 3-3 Examples of Structured Processed Data

Modeler Name	Time	Project Name	General Command	Specific Command	Revit Version	View Type
modeler	10:30:07	StdUnion	"Internal"	"Show or hide recent files	2014	Floor Plan
modeler	10:33:50	StdUnion	"StartupPage"	"Open an existing project	2014	Floor Plan
modeler	10:36:51	StdUnion	"Internal"	"Print the active window	2014	Sheet
modeler	10:45:25	StdUnion	"Internal"	"Activate this viewport	2014	Sheet
modeler	10:48:04	StdUnion	"Internal"	"Modify view templates	2014	Sheet
modeler	10:48:07	StdUnion	"Internal"	"Manage links	2014	Sheet
modeler	10:48:09	StdUnion	"AccelKey"	"Steering wheels	2014	3D View

3.3.2 Input Preparation

The records from five modelers, which each contained over 105,000 log entries, were used to conduct further analyses. The selected modelers designed the interior systems of healthcare building projects conducted in 2013 and 2014. To prepare the input for the GST, the log entries for each user were modeled as long strings of characters. To this end, all available sequences for each modeler were first retrieved and organized. Sequences that had two consecutive commands that were executed 10 minutes or more apart were split and treated as separate series. The 10-minute interval was chosen per the designers' suggestion, since no clear indicator could be found for distinguishing consecutive sequences in the recorded data. Once all command vectors for each user were constructed, the entries were transformed into strings by assigning a unique character to each general command or specific command combination (Figure 3-4). The final strings that were generated for each modeler maintained the original chronological order of the data.

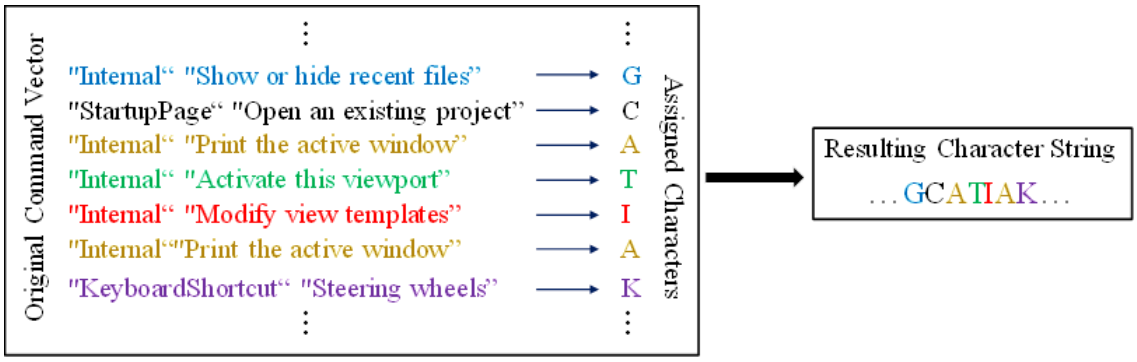

Figure 3-4 Command Vector Transformation

3.3.3 Pattern Extraction

Testing whether a sequence occurs frequently in large databases needs to be performed in an efficient manner. Generalized suffix tree data structures can provide linear time solutions to challenging string mining problems (Gog et al. 2014). Figure 3-5 lists the steps taken to identify frequent command execution sequences where the original order of the data is maintained. Once the input strings for each user were prepared, GST data structures were constructed by considering all possible suffixes. Then, a depth-first search (DFS) was conducted to generate an ordered list of leaf nodes of the tree. Therefore, the leaves corresponding to each internal node are a consecutive sub-list of this ordered list of leaves. The DFS saves the start and end positions of the leaves, based on their DFS order, for each pattern (i.e., internal node) in a helper hash table. The subtraction of the end position and the start position will provide the number of repetitions of the specific pattern. Additionally, in this way, all the instances of that pattern in the original string can be accessed by retrieving the leaf nodes and their corresponding suffix indexes. This technique enables one to not only calculate the frequency of design patterns but also efficiently measure the average time of each pattern in the whole string of commands. These patterns,

then, were filtered based on their minimum length and minimum frequency. At the end of this step, a limited number of the most frequent substrings, with lengths of 3 to 8 that were common between all users, was selected for further analysis. These substrings are called "primitives" as they represent different modeling tasks. Finally, the average time that each user spends performing each primitive was calculated, and the modelers were ranked accordingly.

Input: *a csv file containing recorded entries, minimum length of the extracted sequences l, minimum support (minimum frequency) of the extracted sequences m*
Output: *common command patterns*
01- Initialize Array List L
02- Construct input vectors for designer
03- Build generalized suffix tree considering all input strings
04- for each leaf node f
05- current node ← f
06- while root is not reached
07- p ← parent (current node)
08- Add string id of the leaf node f to the node p if this id has not already been added
09- if the number of string ids added to p ≥ m and ||path(p, root)||>l then
10- add(path(p, root)) to L
11- end if
12- current node ← p
13- end while
14- end for
15- return L

Figure 3-5 Pattern Extraction Using a GST

3.3.4 Discussion of Results

Table 3-4 provides some general information about the data utilized in this chapter. The five modelers selected for further analysis had a total of 582,887 entries. Prior to mining the dataset, some preliminary analyses were conducted to identify the most frequently executed individual commands, and the results are presented in Table 3-5.

“Move selected objects or their copies”, “align references”, and “create a line” were consistently the three most frequently executed individual commands among all five modelers. This consistency among the identified commands further supported the author’s initial hypothesis that there are common command execution patterns among modelers. “Floor plan”, “sheet”, and “3D view” were also identified as the three most frequently used views during the Revit sessions.

Table 3-4 Dataset Statistics

Data set	Period (Year)	Number of entries
Modeler 1	2013	126,815
Modeler 2	2014	122,813
Modeler 3	2014	114,290
Modeler 4	2013	112,082
Modeler 5	2014	106,887

Table 3-5 Most Frequent Individual Commands

	Frequency		
Modeler	Move selected objects or their copies	Align references	Create a line
Modeler 1	19.30%	11.92%	7.89%
Modeler 2	15.27%	13.56%	14.23%
Modeler 3	18.19%	6.21%	11.07%
Modeler 4	15.57%	15.98%	13.07%
Modeler 5	14.31%	6.94%	10.02%

In the next step, frequent command execution patterns for each modeler were retrieved using GST data structures. Several arbitrary minimum support values were tested, of which 250 and 500 were found to be optimal. The minimum length of extracted patterns was also set to 3. The top frequent patterns obtained for the two minimum support values are presented in Table 3-6. The primitives that were extracted for a minimum support of 250 are longer, and they represent meaningful modeling activities. Pattern 1 corresponds

to the task of creating and extending multiple lines. In this case, modelers have hidden a few objects to gain easier access to the elements they want to modify. The second pattern appears to present cases where modelers make copies of different elements and visualize their dimensions. The third pattern captures the commands used to make copies of a specific object in the model and modify them.

Table 3-6 Common Command Sequences

	Pattern 1	Pattern 2	Pattern 3
Extracted Pattern 250	1. Select objects to modify 2. Hide selected elements 3. Create a straight detail line or a detail arc 4. Rotate selected object(s) 5. Trim or extend two lines or walls to make a corner	1. Copy the selection and put it on the clipboard 2. Move copies of selected objects 3. Create aligned dimensions	1. Select objects to modify 2. Create an object similar to the selected object 3. Move selected objects or their copies 4. Align references 5. Finish sketch
Extracted Pattern 500	1. Activate this viewport 2. Copy the selection and put it on the clipboard 3. Deactivate the currently active viewport	1. Move selected objects or their copies 2. Move selected objects or their copies 3. Move selected objects or their copies	1. Deactivate the currently active viewport 2. Activate this viewport 3. Control the visibility and appearance of objects (applied only in the active view)

The obtained patterns became shorter when the minimum support threshold was increased. This observation was expected, since longer sequences tend to match less frequently. The first pattern corresponds to cases where modelers navigate through different viewpoints to select and copy certain objects. The third pattern also captures the command sequences used to change the visibility of different layers. In contrast to these two patterns, it is not clear what specific activity the second command sequence represents. More examples of such repetitive sequences for both thresholds were observed. The presence of such patterns may be because of the noisy input data or the consecutive

execution of similar commands by modelers. Increasing the minimum support beyond 500 resulted in capturing more patterns with repetitive entries that were not representative of any specific modeling tasks.

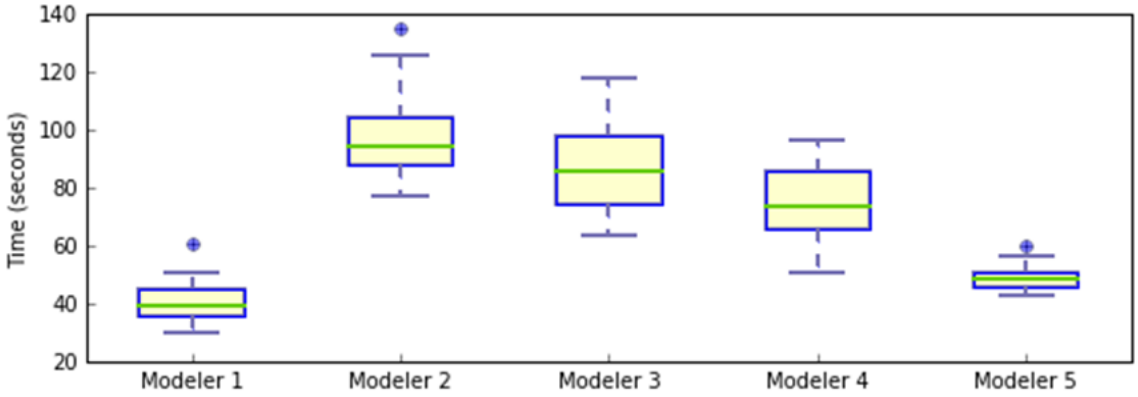

Figure 3-6 Task Execution Times

In addition to identifying the tasks that correspond to different command sequences, the average time it took modelers to execute these tasks was utilized to evaluate their performance. Thirty common patterns with lengths of between 3 and 8 were selected for this purpose. As illustrated in Figure 3-6, there is a visible difference among modelers in terms of the time it takes them to conduct similar modeling tasks. The point outliers measured for modelers 1, 2, and 5 could represent specific tasks that these modelers are particularly less productive in executing and for which they may need further training. A one-way ANOVA test was conducted to empirically check the initial research hypothesis formulated as follows:

> **H_0.** There is no difference among BIM modelers in terms of the average time it takes them to execute similar modeling tasks.

H_a. There is a significant difference among BIM modelers in terms of the average time it takes them to execute similar modeling tasks.

The results are listed in Table 3-7 and

The obtained results are listed in Table 3-9. The calculated *P-values* indicate that, in most cases, the null hypothesis can be rejected in favor of the alternative one with a 99% level of confidence. The test result was not significant for modelers 2 and 3; this can be explained given the fact that modelers 2, 3, and 4 were less experienced in working with Revit compared to others. These findings indicate that the proposed method can capture performance differences and similarities among each pair of modelers when their times for the same type of patterns are compared.

Table 3-8. The calculated *F*-ratio and *P*-value demonstrate that the null hypothesis can be rejected in favor of the alternative one with 99% confidence. Therefore, there is enough evidence to claim that the information embedded in design log files in general, and Revit journals in particular, can effectively capture performance variations among modelers. Another interesting observation is that the average times for the faster modelers has less variation compared to others. For instance, the calculated standard deviations for modelers 1 and 5 (7.4 and 4.4 seconds respectively) are considerably smaller than for the other three modelers (22.6, 15.8, and 12.2 seconds respectively). This could be because faster modelers are more skilled in executing different tasks and thus consistently quicker across the board. However, slower modelers are more skilled in executing some tasks and less experienced in executing others. Therefore, there is a larger variance in terms of the average times it takes them to conduct different modeling tasks.

The author also conducted multiple unpaired student *t*-tests to verify whether the calculated times capture the variations among individual modelers. The null and alternative hypotheses for these tests were formulated as follows:

H_0. $\mu\_1 = \mu\_2$, where μ_1 and μ_2 are the mean of the average times for the two populations being tested.

H_a. $\mu\_1 \neq \mu\_2$.

Table 3-7 Pattern Execution Times

	Average Execution Time (s)				
Patterns (Minimum Support = 250)	**Modeler 1**	**Modeler 2**	**Modeler 3**	**Modeler 4**	**Modeler 5**
1	50.4	99.1	72.7	50.4	51.3
2	37.3	77.7	64.9	75.5	43.0
3	33.3	93.3	63.4	60.6	56.6
4	38.4	95.9	71.4	72.6	51.3
5	38.1	111.6	87.0	96.2	48.2
6	50.7	195.8	138.1	93.5	50.5
7	34.5	88.2	83.5	77.3	50.1
8	37.2	110.3	92.5	90.4	59.6
9	36.7	135.4	77.4	55.5	48.7
10	35.6	92.1	81.6	88.4	56.7
11	33.2	102.8	69.8	74.8	45.3
12	48.3	114.6	100.5	70.2	49.5
13	44.3	95.7	87.3	66.7	56.9
14	48.1	104.8	99.3	89.3	44.4
15	45	121.1	104.7	72.1	46.2
16	31.5	83.4	73.2	63.3	50.4
17	35.2	88.2	99.0	67.6	43.3
18	42.7	93.5	93.1	76.4	48.0
19	29.9	89.4	99.8	61.1	47.2
20	44.1	101.9	96.2	83.0	50.8
21	29.6	92.4	73.4	88.4	52.7
22	60.9	92	105.9	63.7	45.5
23	48.8	80.1	102.3	57.7	50.5
24	47.3	83.3	92.4	70.2	44.2
25	37.9	77.3	66.2	65.1	48.9

26	30.2	80	96.3	75.5	47.4
27	39.7	81.5	79.9	87.0	51.8
28	42.3	109.7	84.2	74.5	43.7
29	44.3	97.6	82.5	66.7	49.2
30	43.9	102	91.3	90.9	43.1
$\bar{X}$	40.666	99.694	87.666	74.164	49.184
s	7.419	22.644	15.794	12.231	4.363
$\bar{X}_{ave}$	70.275				

The obtained results are listed in Table 3-9. The calculated *P*-values indicate that, in most cases, the null hypothesis can be rejected in favor of the alternative one with a 99% level of confidence. The test result was not significant for modelers 2 and 3; this can be explained given the fact that modelers 2, 3, and 4 were less experienced in working with Revit compared to others. These findings indicate that the proposed method can capture performance differences and similarities among each pair of modelers when their times for the same type of patterns are compared.

Table 3-8 The *F*-test Results

Source	df*	SS**	MS***	F-statistic	P-value
treatments	4	75137.236	18784.309	95.2698	0.0000
error	145	28589.583	197.170		
total	149	103726.818			

* Degree of freedom
** Sum of squares
*** Mean square

Table 3-9 Pairwise Comparison of Modelers

	Modeler 2	Modeler 3	Modeler 4	Modeler 5	
Modeler 1	0.00000	0.00000	0.00001	0.00001	***T*-test *P*-Values**
Modeler 2		0.02029	0.00000	0.00000	
Modeler 3			0. 00048	0.00000	
Modeler 4				0.00001	

3.4 Chapter Summary

Design modeling is crucial to the success of construction projects. However, a high-quality building model does not occur in a void; instead, it is the result of a well-managed, multidisciplinary process. To effectively manage such processes, accurate performance data are required to evaluate and track different metrics. In this chapter, the utilization of modeling development information, embedded in design log files produced by Autodesk Revit, was proposed as a rich source of performance data. To this end, the necessary steps to extract and analyze the data were outlined, and an effort was made to make a contribution at each step. Throughout the first step, the format in which different information items are stored was identified. Using these protocols, a text parser was developed to extract the required information items. This text parser accepts the raw journal files and produces structured CSV files. The obtained data were further cleaned and organized by removing non-value-adding entries, such as cancel and error messages. The conclusion derived at this level is that using the suggested approach will help to process unstructured journal log files and produce high-quality input data for the mining algorithm. In the next step, GST data structures were utilized to identify common command sequences among Revit users. First, command sequences were transformed into character-based input strings. Then, the transformed data were utilized to construct a GST. Frequent command patterns were identified by conducting a DFS on the constructed trees, and the extracted patterns for different users were compared against each other to identify shared sequences. The conclusion at this step is that a GST-based string mining approach is an efficient method for extracting common command patterns among several modelers. In addition to identifying these common patterns, the average time it takes the selected modelers to

execute command sequences was calculated. The obtained results indicate that there is a statistically significant difference among the modelers in terms of the time it takes them to conduct similar modeling tasks. This finding confirms the initial hypothesis that Revit journal files can be used as a rich source of data to capture performance variations among multiple modelers.

This chapter contributes to the state of knowledge by proposing a tailored string mining algorithm that is capable of extracting meaningful information from timestamped design development data. Furthermore, the proposed methodology contributes to the state of practice by enabling design project managers to gain unprecedented insight into the evolution of a building model using the information embedded in design log files.

CHAPTER 4: AUTOMATIC EXTRACTION OF MODELER PERFORMANCE INFORMATION

4.1 Introduction

It was demonstrated that design log files can be used to collect objective data that are necessary to measure, benchmark, and compare design modeling performance. However, there is still a need to use text parsers to retrieve information from log files, given that the data embedded in them are unstructured. This chapter addresses the second research objective of this dissertation, which is to directly extract modeling information from design software solutions using APIs. To achieve this objective, the remainder of this chapter is structured as follows: first, a brief introduction to object-oriented software packages is presented; then, the proposed API-based data collection framework and the steps conducted in this chapter to implement it are described; thereafter, the plugin developed in this chapter to automatically collect design modeling performance information from Autodesk Revit is explained; and finally, the conclusions are presented.

This chapter contributes to the existing body of knowledge by introducing a framework for capturing accurate performance data from design software solutions. The proposed methodology can be implemented in any design software package with open-access capability. The findings of this chapter can help design managers to acquire progress information without the need to manually record and report data.

4.2 Application Programming Interfaces

Conducting data-driven design monitoring requires massive amounts of granular information to be collected from various sources and several project participants, and the emergence of API-enabled design software packages provides a unique opportunity to extend the core capabilities of VDC modeling tools (Zhang et al. 2013). Even though such APIs can also be utilized to record design development events and UI interactions with design models, no research has been conducted that uses such capabilities to monitor design progress. In simple words, various members of the design team leave traces behind as they interact with virtual design models. Historical records of design development events (for example, changes in design elements, and executed commands by each user) provide a rich source of information about the progress of a design project.

The motivation behind the proposed approach is that design software protocols, such as APIs, can be devised to create a novel data acquisition method. Once the system is placed in a design environment, a wide range of design development events (for example, attributes related to the performed design tasks, UI interaction features, and object parameters) are captured in real time and automatically transferred to a database of information requirements for design monitoring. This capability can facilitate the creation of a non-intrusive mechanism to capture model development events and the parametric information of elements throughout different phases of a design project. This chapter is the first attempt to introduce the use of design software APIs as an alternative source of information for monitoring design progress and analyzing performance. The fundamentally different property of the proposed approach is its adaptability to the evolution of design.

4.3 Research Description

In this research, the functionality of event handling that is present in design software protocols is utilized to record model development events. As noted in the introduction section, collecting objective data is necessary for evaluating design performance metrics. Therefore, the first step is to create a robust framework to automatically capture detailed design progress information. Event handlers offer a computationally efficient solution to address the data collection issue: an event-handling-based system can record all possible software actions ranging from creating, selecting, and modifying elements to navigating through different zones of a virtual model, as well as information regarding the current dimension, cost, material, and family features of each element.

The Autodesk Revit API allows for the development of external applications. This framework allows one to customize Revit ribbon panels and controls, and record model development events (Autodesk 2017a). This includes more than 1,500 events ranging from dialog box showing to errors and ribbon button clicks. An event-based framework can be implemented on parametric design software solutions other than Revit, since these solutions are created based on the principles of object-oriented programming (OOP).

As illustrated in Figure 4-1, three different event handler functions were used to develop a data collector Revit plugin. The first function records the user's interaction with Revit ribbon buttons, and it is triggered each time modelers use the Revit interface or a keyboard shortcut. The second function reacts whenever a model element is added, modified, or deleted. In addition to recording the type of change, other information items, such as the element's type, its name, and its globally unique identifier (GUID), are

collected. The final function is triggered when an error occurs during the modeling session. This includes memory and user-induced errors, and any other software failure that interrupts the workflow. The information captured using these functions is written to a text file in real time. Also, the collected data are recorded in a pre-defined, comma-separated format that is machine-readable. This facilitates the analysis of modeling performance data, since there is no need for a text parser to retrieve the information. The functionality of this plugin is validated through experiments that are conducted in the next chapter, and the steps taken to implement this framework are outlined in the following section.

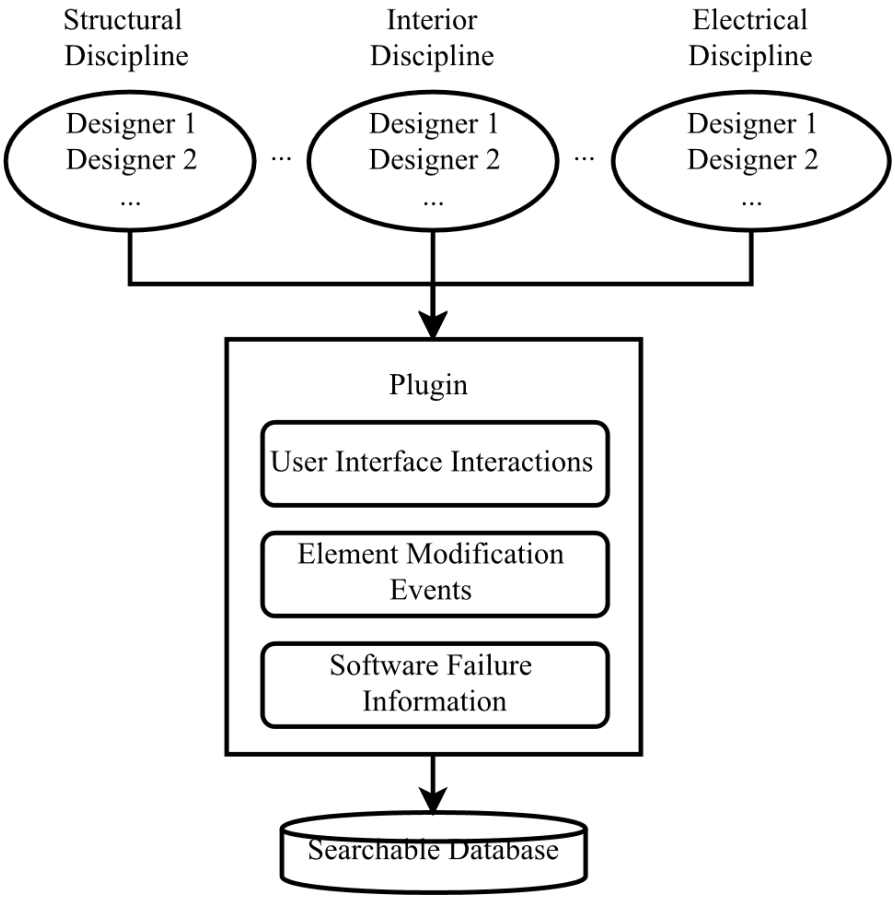

Figure 4-1 Research Methodology for Developing a Data Collector Plugin

4.4 Implementation

4.4.1 External Application Interface

To develop a Revit plugin, the IExternalApplication interface should be implemented. This interface has two abstract methods: OnStartup() and OnShutdown(). These methods, which should be overridden, are called when Revit starts and closes respectively. Registering event handlers in the OnStartUp() method enables the plugin to automatically start working when a Revit project is opened. This is particularly important as the goal of this chapter is to collect modeling progress data without any manual intervention.

Figure 4-2 presents the segment of C# code where the OnStartUp() and OnShutDown() methods are implemented. The input parameter, UIControlledApplication, provides access to the group of event handlers that needs to be activated, and Revit periodically checks whether any registered events are raised. The following three event handlers were registered in the OnStartUp() method:

- ***RibbonItemExecutedEventArgs***—this event handler is triggered when a ribbon button is clicked or a keyboard shortcut is used. ID_EDIT_MOVE and ID_OBJECTS_WALL_RibbonListButton, which can be used to move elements and create a wall respectively, are instances of such events.
- ***DocumentChangedEventArgs***—this event handler is executed when a model element is changed. These changes include addition, any type of modification, or deletion.

```
public class RecordCommands : IExternalApplication
{
    public static RecordCommands thisApp = null;
    /// Implement this method to subscribe to event.

    public Result OnStartup(UIControlledApplication application)
    {
        thisApp = this;
        try
        {
            // Register event.
            ComponentManager.ItemExecuted += new EventHandler
                <RibbonItemExecutedEventArgs>(CommandExecuted);
            application.ControlledApplication.DocumentChanged += new EventHandler
                <DocumentChangedEventArgs>(DocumentChangeTracker);
            application.ControlledApplication.FailuresProcessing += new EventHandler
                <FailuresProcessingEventArgs>(FailureTracker);
            string tempFolder = "C:\\Users\\Saman\\Dropbox\\Add-in\\EventTester\\Outputs";
            string outputFile = Path.Combine(tempFolder, "Project.txt");
            using (StreamWriter sw = new StreamWriter(outputFile, true))
            {
                DateTime now = DateTime.Now;
                sw.WriteLine("Revit started up at " + now + ".");
            }
        }

        catch (Exception)
        {
            return Result.Failed;
        }

        return Result.Succeeded;
    }


    public Result OnShutdown(UIControlledApplication application)
    {
        // remove the event.
        application.ControlledApplication.DocumentChanged -= DocumentChangeTracker;
        application.ControlledApplication.FailuresProcessing -= FailureTracker;
        ComponentManager.ItemExecuted -= CommandExecuted;

        string tempFolder = "C:\\Users\\Saman\\Dropbox\\Add-in\\EventTester\\Outputs";
        string outputFile = Path.Combine(tempFolder, "Project.txt");
        using (StreamWriter sw = new StreamWriter(outputFile, true))
        {
            DateTime now = DateTime.Now;
            sw.WriteLine("Revit shut down at " + now + ".");
        }

        return Result.Succeeded;
    }
```

Figure 4-2 Event Handler Registration in OnSartUp() and OnShutdown() Methods

- ***FailuresProcessingEventArgs*—**this event handler is activated when a user-induced error or software failure occurs, including memory errors and wrong element positioning warnings.

Once these events are registered at the start up, the plugin writes the current time and date to a text file where all information will be recorded. This task is done by creating a DateTime object, which returns the current date and time. The OnShutdown() method is called when Revit is closed, and as evident in Figure 4-2, this method deactivates the event handlers that are registered in OnStartup(). Unregistering events is necessary to avoid interruption caused by the plugin producing exceptions. Similar to the previous method, OnShutdown() uses a DateTime object to record the current date and time in the destination text file.

4.4.2 Ribbon Item Event Handler

Ribbon item event handler is a ribbon-specific interface under the Autodesk.Windows class. This class provides access to track the execution of commands using ribbon buttons and keyboard shortcuts in several Autodesk products, including Revit. Figure 4-3 presents the implementation of the CommandExecuted class, which responds to the signals raised by the RibbonItemExecutedEventArgs event handler. Once this event is raised, the plugin writes the following information regarding the nature of the executed command to the text file:

- ***Entry identifier***—the keyword "Command" is used in the beginning to specify the type of information recorded.
- ***Current time and date (now)***—these are recorded using a DateTime object.

- ***Command-execution point (it.ToString())***—this specifies whether the command is executed using ribbon buttons or keyboard shortcuts (for example, "UIFramework.SketchGalleryItem" and "Autodesk.Windows.RibbonButton")
- ***Command id (it.Id)***—this represents the unique IDs specified by Autodesk (for example, "ID_New_Revit_Design_Model" and "ID_Edit_Move_Copy").
- ***Complementary command information (it.Cookie)***—this contains more detailed information regarding the nature of executed commands (for example, "SketchGalleryItem_ID_Object_3D_Curve_Spline_Through_Points" and "ID_Button_Select_Modify").

```
private void CommandExecuted(object sender, RibbonItemExecutedEventArgs args)
{
    try
    {
        Autodesk.Windows.RibbonItem it = args.Item;
        string tempFolder = "C:\\Users\\Saman\\Dropbox\\Add-in\\EventTester\\Outputs";
        string outputFile = Path.Combine(tempFolder, "Project.txt");


        if (args != null)
        {
            using (StreamWriter sw = new StreamWriter(outputFile, true))
            {
                DateTime now = DateTime.Now;
                sw.WriteLine("Command, " + now + ", " + it.ToString() + ", " +
                    it.Id + ", " + it.Cookie);
            }
        }
    }

    catch (Exception ex)
    {
        Autodesk.Revit.UI.TaskDialog.Show("Add-In Fail", ex.Message);
    }
}
```

Figure 4-3 Ribbon Item Event Handler Implementation

4.4.3 Document Changed Event Handler

Many actions, such as dragging and deleting elements, are conducted without any interactions with the Revit ribbon buttons and keyboard shortcuts. In these cases, the ribbon item event handler will not receive any signals, and it will not write the event to the destination text file. To address this issue, the action DocumentChangedEventArgs is activated when any change occurs to the model, including the execution of non-ribbon commands.

```
// Records changes made to elements, modifications, additions of elements
public void DocumentChangeTracker(object sender, DocumentChangedEventArgs args)
{
    Application app = sender as Application;
    UIApplication uiapp = new UIApplication(app);
    UIDocument uidoc = uiapp.ActiveUIDocument;
    Document doc = uidoc.Document;
    View currentView = uidoc.ActiveView;

    // The file path where information are recorded
    string user = doc.Application.Username;
    string filename = doc.PathName;
    string filenameShort = Path.GetFileNameWithoutExtension(filename);
    string tempFolder = "C:\\Users\\Saman\\Dropbox\\Add-in\\EventTester\\Outputs";
    string outputFile = Path.Combine(tempFolder, "Project.txt");

    // Elements that are modified or added are identified here
    Selection sel = uidoc.Selection;
    ICollection<ElementId> deletedElements = args.GetDeletedElementIds();
    ICollection<ElementId> changedElements = args.GetModifiedElementIds();
    ICollection<ElementId> addedElements = args.GetAddedElementIds();
    ICollection<ElementId> selectedIds = sel.GetElementIds();
    int counter = deletedElements.Count + changedElements.Count + addedElements.Count;
```

Figure 4-4 Document Changed Event Handler Implementation

Figure 4-4 presents the implementation of the DocumentChanged event handler. Three different element modifications are of interest:

- ***Element deletion***—the GUIDs of deleted elements are collected using the "GetDeletedElementIds()" method.

- ***Element modified***—the GUIDs of existing elements that have been modified in any form are collected using the "GetModifiedElementIds()" method.
- ***Element addition***—the GUIDs of newly added elements are collected using the "GetAddedElementIds()" method.

Figure 4-5 presents the implementation of the element deletion event handler. The current date and time are returned using a DateTime object. Furthermore, in case there are any deleted elements, the program loops through the "DeletedElements" collection. The following information is then printed for each deleted element:

- ***Entry identifier***—the keyword "ElementChange" displays the type of information recorded.
- ***User***—this is the name of the modeler.
- ***Current time and date (now)***—these are the current date and time captured by a DateTime object.
- ***Change identifier—***this represents the keyword "Deleted" to specify the type of change.
- ***Element id (id.IntegerValue)***—this is the deleted element's GUID.

```
// Writing information to the text file
using (StreamWriter sw = new StreamWriter(outputFile, true))
{
    DateTime now = DateTime.Now;


    // Deleted elements are recorded
    // ElementChange, user, time, Deleted,
    if (deletedElements.Count != 0)
    {
        foreach (ElementId id in deletedElements)
        {
            sw.WriteLine("ElementChange, " + user + ", " + now +
                ", Deleted, " + id.IntegerValue);
        }
    }
```

Figure 4-5 Element Deleted Event Handler Implementation

The element addition function is implemented as demonstrated in Figure 4-6. An advantage of this method is its access to BoundingBoxXYZ objects. The bounding box objects that are obtained from elements that represent "the boundary of the element in a given view" (Revit API Doc 2016). The extents of the bounding box are specified by "three orthogonal planes extended through the minimum (Min) and maximum (Max) points." This box coordination information is useful for identifying the initial position of elements when they are added to a model. Furthermore, the information items recorded when an element is added to a model are as follows:

- ***Entry identifier***—this is the keyword "ElementChange" to identify the type of information entry.
- ***User***—this is the name of the modeler.
- ***Current date and time (now)***—these are the current date and time captured by a DateTime object.
- ***Change identifier***—this represents the keyword "ADDED" to specify the type of change.

- ***Element category (Category.Name)***—this identifies the category or subcategory to which an element belongs, for example, doors or walls.
- ***Element bounding box min (box.Min)***—this is the minimum extent point of the element bounding box.
- ***Element bounding box max (box.Max)***—this is the maximum extent point of the element bounding box.
- ***Element type (Element.GetType)***—this provides more details on the element category.
- ***Element Family (Element.Name)***—this is the element family that is defined as a "group of elements with a common set of properties, called parameters, and a related graphical representation" (Revit API Doc 2017).
- ***Element id (id.IntegerValue)***—this is the modified element's GUID.
- ***Current view (currentView.Name)***—this is the current elevation perspective of the model.
- ***Project name (doc.PathName)***—this represents the directory address of the project being modeled.

```
if (addedElements.Count != 0)
{
    foreach (ElementId id in addedElements)
    {
        BoundingBoxXYZ box = doc.GetElement(id).get_BoundingBox(null);
        sw.WriteLine("ElementChange, " + user + ", " + now + ", ADDED, " +
            doc.GetElement(id).Category.Name + ", " + box.Min.ToString() +
            ", " + box.Max.ToString() + ", " + doc.GetElement(id).GetType().Name +
            " , " + doc.GetElement(id).Name + ", " + id.IntegerValue +
            ", " + currentView.Name + ", " + doc.PathName);
    }
}
```

Figure 4-6 Element Addition Event Handler Implementation

Finally, the implementation of the element modified method is illustrated in Figure 4-7. This method is triggered either when a ribbon item button is clicked or when a direct element modification, such as dragging, takes place. In case of ribbon buttons, both the ribbon item event handler and the element modified functions will produce entries. The information that the element modified method records is as follows:

- ***Entry identifier***—this is the keyword "ElementChange" to identify the type of information entry.
- ***User***—this is the name of the modeler.
- ***Current date and time (now)***—these are the current date and time captured by a DateTime object.
- ***Change identifier***—this represents the keyword "MODIFIED" to specify the type of change.
- ***Element category (Category.Name)—***this identifies the category or subcategory to which an element belongs, such as doors or walls.
- ***Element bounding box min (box.Min)—***this denotes the minimum extent point of the element bounding box.
- ***Element bounding box max (box.Max)—***this denotes the maximum extent point of the element bounding box.
- ***Element type (Element.GetType)—***this provides more details on element the category.
- ***Element Family (Elemeent.Name)—***this is the element family.
- ***Element id (id.IntegerValue)—***this represents the modified element's GUID.

- ***Current view (currentView.Name)*—**this is the current elevation perspective of the model.
- ***Project name (doc.PathName)*—**this is the directory address of the project being modeled.

```
if (changedElements.Count != 0)
{
    foreach (ElementId id in changedElements)
    {
        if (selectedIds.Contains(id))
        {
            BoundingBoxXYZ box = doc.GetElement(id).get_BoundingBox(null);
            sw.WriteLine("ElementChange, " + user + ", " + now +
                ", MODIFIED, " + doc.GetElement(id).Category.Name +
                ", " + box.Min.ToString() + ", " + box.Max.ToString() +
                ", " + doc.GetElement(id).GetType().Name + " , " +
                doc.GetElement(id).Name + ", " + id.IntegerValue +
                ", " + currentView.Name + ", " + doc.PathName);
        }
    }
}
```

Figure 4-7 Element Modification Event Handler Implementation

4.4.4 Software Failure Event Handler

Figure 4-8 presents the implementation of the software failure event handler. Events, such as errors and warnings, signal this handler to capture the following information:

- ***Entry identifier***—the keyword "ElementChange" to identify the type of information entry.
- ***User***—the name of the modeler.
- ***Current date and time (now)***—the current date and time captured by a DateTime object.

- ***Failure type (Failure.GetTransactionName)***—the names of the transactions associated with the failure event.

```
// Records errors and writes them to the Project.txt file
private void FailureTracker(object sender, FailuresProcessingEventArgs e)
{
    Application app = sender as Application;
    UIApplication uiapp = new UIApplication(app);
    UIDocument uidoc = uiapp.ActiveUIDocument;
    Document doc = uidoc.Document;

    // The file path where information are recorded
    string user = doc.Application.Username;
    string filename = doc.PathName;
    string filenameShort = Path.GetFileNameWithoutExtension(filename);
    string tempFolder = "C:\\Users\\Saman\\Dropbox\\Add-in\\EventTester\\Outputs";
    string outputFile = Path.Combine(tempFolder, "Project.txt"); // filenameShort
    // + "_Errors.txt");

    // inside event handler, getting all warnings
    FailuresAccessor failuresAccessor = e.GetFailuresAccessor();
    IList<FailureMessageAccessor> fmas = failuresAccessor.GetFailureMessages();

    // Writing information to the text file
    using (StreamWriter sw = new StreamWriter(outputFile, true))
    {
        DateTime now = DateTime.Now;
        if (fmas.Count != 0)
        {
            foreach (FailureMessageAccessor fma in fmas)
            {
                sw.WriteLine("Error, " + user + ", " + now + ", " +
                    failuresAccessor.GetTransactionName());
            }
        }
    }
}
```

Figure 4-8 Software Failure Event Handler Implementation

4.5 Plugin Functionality Validation

A test was conducted to ensure the functionality of the developed Revit plugin, and through this test, simple 3D models were created. Every ribbon command execution, direct element modification (mouse drag), keyboard shortcut, and error warning were manually recorded, and these recordings were then compared with the output produced by the plugin (Figure 4-9). The accuracy of the plugin's performance was validated by comparing the output with the manually recorded information.

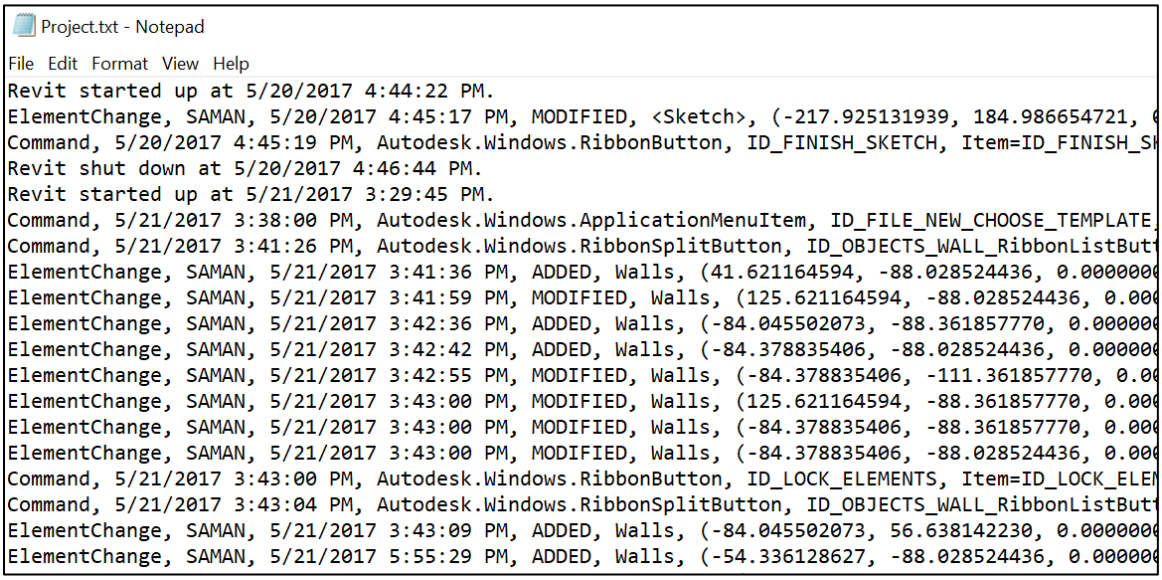

Figure 4-9 Sample Plugin Output

4.6 Chapter Summary

This chapter described a non-intrusive mechanism to capture model development events that documents the evolution of design throughout different phases of a project. Current design practices rely on VDC tools to generate and manage digital representations of the physical and functional characteristics of a project. The major focus of the proposed

approach is to capture temporal modeling events using software application protocols. This novel approach provides practitioners with broader access to granular design development data that can be used to generate insights into design modeling performance patterns.

This chapter utilized modeling solution APIs to automatically collect and store timestamped design development information. The proposed passive data recording approach allows for the real-time capture of comprehensive UI interactions and model element modification events. These recordings consist of all possible software actions, ranging from creating, selecting, and modifying elements to navigating through different zones of a virtual model. Such recordings, constituting timestamped event sequences, are organized and stored in machine-readable formats: .txt files.

The presented system can collect and consolidate data from multiple modelers who are simultaneously working on different design models. This capability allows one to obtain information from the different parties involved in a design team. Also, this system is designed in such a way that it can distinguish between projects that a user might be working on concurrently. The efficiency achieved by using event handlers in the developed method allows it to be implemented on different platforms. Additionally, disruptions to the modeling operation are avoided by recording information during the idle time of the software solutions.

The proposed framework was implemented as an Autodesk Revit plugin, and an experiment was conducted to verify the accuracy of this plugin. Throughout this experiment, manual recordings of model development events were compared against the automatically generated plugin output. This section contributes to the state of knowledge by introducing a framework for automatically capturing accurate performance data from

design software solutions, and it contributes to the state of practice by helping design managers to acquire progress information without the need to manually record and report data.

CHAPTER 5: FINDING OPTIMAL MODELING TEAM CONFIGURATION

5.1 Introduction

The final objective of this study seeks to utilize the proposed design modeling performance data collection approach to identify the optimal modeling team configuration. As described in the methodology section, an experiment was conducted to collect modeling performance data using the Revit plugin developed in chapter 4. The collected data were then analyzed to evaluate participants' modeling performance. The EDD priority rule was then used in combination with the CPM to calculate the expected lateness for different configurations of the modeling team. The remainder of this chapter is structured as follows: first, the details of the Revit plugin experiment are outlined; then, the findings of the analysis that was conducted on the collected data are presented; and, the obtained results are utilized to assess the performance of different team conjugations. Furthermore, a summary of the chapter is presented in the final section.

5.2 Experiment Description

In this experiment, five Master of Architecture students at the Georgia Institute of Technology were asked to produce 3D models of a youth and family center building using Autodesk Revit. Full sets of production drawings, including the floor plan, roof plan, and building elevations (Figure 5-1 and Figure 5-2), were provided by an Atlanta-based architectural company. The participants utilized a computer for modeling, with the Revit plugin installed to track and record model development events.

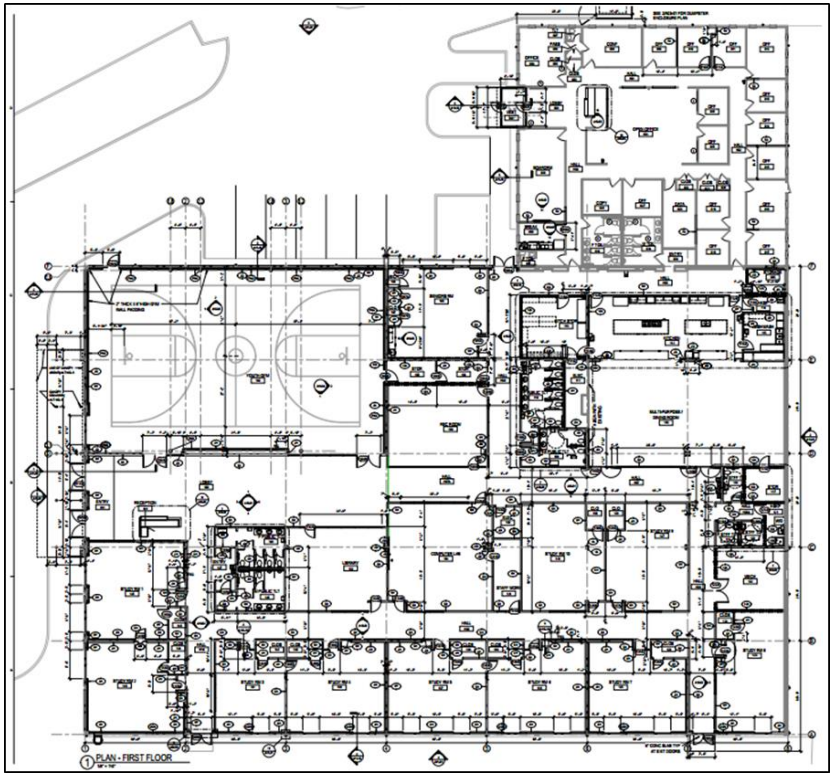

Figure 5-1 Building Floor Plan

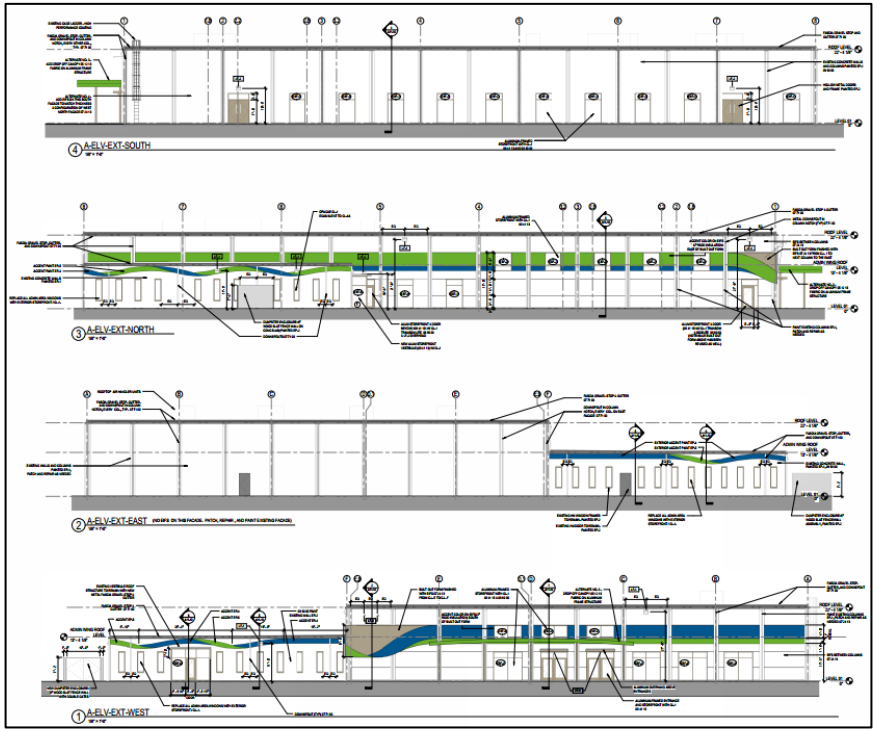

Figure 5-2 Building Elevations

The experiment focused on specific activities to model walls and windows within the overall process. Participants were asked to model the building to an LoD that accurately reflected the information provided in the initial drawings, and the produced models were checked to ensure that LOD 200 (AIA 2013) requirements were met. Figure 5-3 is an example of a 3D model that one of the modelers created. The data collected throughout the experiment were analyzed to evaluate different modeling performance measures. The results of the analysis are presented next.

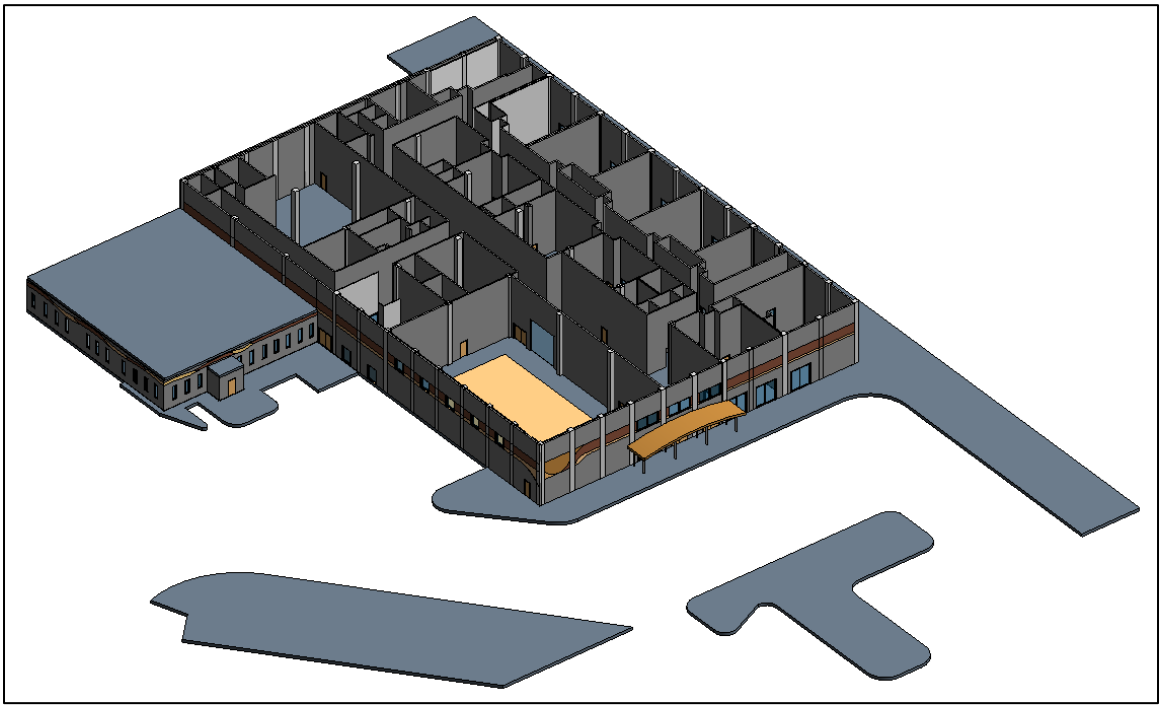

Figure 5-3 Sample 3D Model Produced by Experiment Participants

5.3 Analysis of Experiment Data

5.3.1 Processing Collected Modeling Data

The Python programming language was used to organize and store the collected modeling data. In particular, the pandas software library (pandas 2017), which is written for data manipulation and analysis, was utilized for the following reasons:

- The pandas library has an R-style data frame structure, which allows column names and indexing; this is helpful for keeping track of the data.
- The pandas library has efficient input and output capabilities to read and write data from and to different database formats.
- Pandas' multiple built-in functionalities, such as joins, merges, and searches, make it an effective data-processing tool.
- Pandas can store non-homogeneous data types in the same data frame. This is important as the recorded data include both numeric and alphabetic types.

The processed data are stored in three element changes, executed commands, and errors dataframes. Figure 5-4 depicts the segment of code to initialize these dataframes.

```
import pandas as pd
from datetime import datetime, timedelta

modelers = ["Modeler1", "Modeler2", "Modeler3", "Modeler4", "Modeler5"]
directory = 'C:\\Users\\Saman\\Revit Test Results\\'

df_elem_change = pd.DataFrame({"User": [], "Time": [], "Action":[],
                               "Category":[], "Min":[], "Max":[], "Center":[],
                               "Type": [], "Family": [], "ID": [], "View": [],
                               "Project": []})
df_cmd = pd.DataFrame({"Time": [], "Exec_Pt": [], "Cmd_ID": [],
                       "Cmd_Info": []})
df_err = pd.DataFrame({"User": [], "Time": [], "ErrorType":[]})
```

Figure 5-4 Pandas Data Frame Initialization

The "modelers" array includes the name of all experiment participants, and "directory" is a string variable that contains the directory address where the collected text files are stored. An explanation of the three dataframes' column names can be found in Table 5-1.

Table 5-1 Data Frame Column Names

	Dataframes		
	Element Changes (Deleted, Modified, Added)	Commands	Errors
Column Names	*User*: modeler name *Time*: current date and time *Action*: change identifier *Category*: changed element category *Min*: element bounding box min *Max*: element bounding box max *Center*: element bounding box center *Type*: element type *Family*: element family *ID*: element ID *View*: current view *Project*: project name	*Time*: current date and time *Exec_Pt*: command-execution point *Cmd_ID*: command ID *Cmd_Info*: complementary command information	*User*: modeler name *Time*: current date and time *ErrorType*: failure type

The model development data collected throughout the Revit sessions are stored in predefined, comma-separated text files. This provides the files with a structure that facilitates reading and organizing the obtained data. Figure 5-5 and Figure 5-6 contain the segments of the Python code that process and store command execution and software failure information in commands and errors dataframes respectively. The code loops through all modelers' text files, and the recordings are read line by line. The nature of each recording is specified by reading the entry identifiers ("Command" and "Error"), and the current time and date are also stored as a DateTime object. Finally, all information items

are recorded in command or error data frames under the corresponding columns specified in Table 5-1.

```
for i in range(5):
    fname = directory + modelers[i] + "\\Project.txt"
    counter = 0
    with open(fname) as f:
        for line in f:
            if line[0] != "R":
                content = line.strip().split(',')

                if content[0] == 'Command':
                    dt = content[1].strip()
                    if dt[-2:] == "PM" and dt[-11:-9] != "12":
                        curr_tm = datetime.strptime(dt, "%m/%d/%Y %H:%M:%S %p")
                        + timedelta(hours=12)
                    else:
                        curr_tm = datetime.strptime(dt, "%m/%d/%Y %H:%M:%S %p")
                    data = pd.DataFrame({"Time": [curr_tm],
                    "Exec_Pt": [content[2].strip()],
                    "Cmd_ID": [content[3].strip()],
                    "Cmd_Info": [content[4].strip()[5:]]})
                    df_cmd = df_cmd.append(data, ignore_index=True)
```

Figure 5-5 Storing Command Execution Data in the Commands Data Frame

```
if content[0] == 'Error':
    user = content[1].strip()
    dt = content[2].strip()
    if dt[-2:] == "PM" and dt[-11:-9] != "12":
        curr_tm = datetime.strptime(dt, "%m/%d/%Y %H:%M:%S %p")
        + timedelta(hours=12)
    else:
        curr_tm = datetime.strptime(dt, "%m/%d/%Y %H:%M:%S %p")
    error_tp = content[3].strip()
    data = pd.DataFrame({"User": [user], "Time": [curr_tm],
    "ErrorType":[error_tp]})
    df_err = df_err.append(data, ignore_index=True)
```

Figure 5-6 Storing Software Failure Data in the Errors Data Frame

Figure 5-7 and Figure 5-8 present the Python code segments that deal with reading and storing element change entries, including element deletions, modifications, and additions. Also, the current date and time are stored using a DateTime object. The element center coordination is calculated by taking the average of the x, y, and z elements'

minimum and maximum coordinates. All information items are finally stored in the element changes data frame under related columns.

```
if content[0] == "ElementChange":
    user = content[1].strip()
    dt = content[2].strip()
    if dt[-2:] == "PM" and dt[-11:-9] != "12":
            curr_tm = datetime.strptime(dt, "%m/%d/%Y %H:%M:%S %p")
            + timedelta(hours=12)
    else:
        curr_tm = datetime.strptime(dt, "%m/%d/%Y %H:%M:%S %p")
    #print(curr_tm)
    action = content[3].strip()
    if action == "Deleted":
        data = pd.DataFrame({"User": [user], "Time": [curr_tm],
                             "Action":[action], "ID":[int(content[4].strip())]})
        df_elem_change = df_elem_change.append(data, ignore_index=True)
    else:
        min_x = float(content[5].strip()[1:])
        min_y = float(content[6].strip())
        min_z = float(content[7].strip()[:-1])
        max_x = float(content[8].strip()[1:])
        max_y = float(content[9].strip())
        max_z = float(content[10].strip()[:-1])
        center_x = max_x /2.0 + min_x / 2.0
        center_y = max_y / 2.0 + min_y / 2.0
        center_z = max_z / 2.0 + min_z / 2.0
        if action == "MODIFIED":
            data = pd.DataFrame({"User": [user], "Time": [curr_tm],
                                 "Action":[action], "Category":[content[4].strip()],
                                 "Min":[(min_x, min_y, min_z)],
                                 "Max":[(max_x, max_y, max_z)],
                                 "Center":[(center_x, center_y, center_z)],
                                 "Type": [content[11].strip()],
                                 "Family": [content[12].strip()],
                                 "ID": [int(content[13].strip())],
                                 "View": [content[14].strip()],
                                 "Project": [content[15].strip()]})
            df_elem_change = df_elem_change.append(data, ignore_index=True)
```

Figure 5-7 Storing Deletions and Modifications in the Element Changes Data Frame

```
if action == "ADDED":
    data = pd.DataFrame({"User": [user], "Time": [curr_tm],
                         "Action":[action], "Category":[content[4].strip()],
                         "Min":[(min_x, min_y, min_z)],
                         "Max":[(max_x, max_y, max_z)],
                         "Center":[(center_x, center_y, center_z)],
                         "Type": [content[11].strip()],
                         "Family": [content[12].strip()],
                         "ID": [int(content[13].strip())],
                         "View": [content[14].strip()],
                         "Project": [content[15].strip()]})
    df_elem_change = df_elem_change.append(data, ignore_index=True)
```

Figure 5-8 Storing Addition in the Element Changes Data Frame

While data frames are suitable for storing information, they cannot be directly saved on a computer's hard disk. Therefore, data frames should be written to the hard disk to make the data available for analysis even after the data processing program has finished running. A comma-separated file format was selected to save the data frames, as indicated in Figure 5-9. The produced outputs are presented in Figure 5-10 and Figure 5-11.

```
df_cmd.to_csv("Commands_DataFrame.csv")
df_elem_change.to_csv("Element_Changes_DataFrame.csv")
df_err.to_csv("Errors_DataFrame.csv")
```

Figure 5-9 Writing Data Frame Information to CSV Files

	A	B	C	D	E	F	G	H	I	J	K	L	M
		Action	Category	Center	Family	ID	Max	Min	Project	Time	Type	User	View
0		ADDED	Structural	(0.0, 0.0, 4	2X2	342148	(0.492125	(-0.492125984, -0.49		5/4/2017 15:52	FamilySym	Saman	Level 1
1		ADDED	Structural	(137.6297	2X2	342152	(138.6297	(136.629784797, -75.		5/4/2017 15:52	FamilyInst	Saman	Level 1
2		ADDED	Analytical	(137.629784797, -74.		342153	(137.6297	(137.629784797, -74.		5/4/2017 15:52	AnalyticalI	Saman	Level 1
3		ADDED	Analytical	(137.629784797, -74.		342252	(137.6297	(137.629784797, -74.		5/4/2017 15:52	Hub	Saman	Level 1
4		ADDED	Analytical	(137.629784797, -74.		342253	(137.6297	(137.629784797, -74.		5/4/2017 15:52	Reference	Saman	Level 1
5		ADDED	Analytical	(137.629784797, -74.		342254	(137.6297	(137.629784797, -74.		5/4/2017 15:52	Hub	Saman	Level 1
6		ADDED	Analytical	(137.629784797, -74.		342255	(137.6297	(137.629784797, -74.		5/4/2017 15:52	Reference	Saman	Level 1
7		ADDED	Cameras	(138.8122	{3D}	342313	(140.6369	(136.987556272, -77.		5/4/2017 15:52	Element	Saman	Level 1
8		ADDED	Structural	(137.6297	2X2	342579	(138.6297	(136.629784797, -75.		5/4/2017 15:56	FamilyInst	Saman	Level 1
9		ADDED	Analytical	(137.629784797, -74.		342580	(137.6297	(137.629784797, -74.		5/4/2017 15:56	AnalyticalI	Saman	Level 1
10		ADDED	Structural	(137.6297	2X2	342865	(138.6297	(136.629784797, -42.		5/4/2017 15:56	FamilyInst	Saman	Level 1
11		ADDED	Analytical	(137.629784797, -41.		342866	(137.6297	(137.629784797, -41.		5/4/2017 15:56	AnalyticalI	Saman	Level 1
12		ADDED	Analytical	(137.629784797, -41.		342956	(137.6297	(137.629784797, -41.		5/4/2017 15:56	Hub	Saman	Level 1
13		ADDED	Analytical	(137.629784797, -41.		342957	(137.6297	(137.629784797, -41.		5/4/2017 15:56	Reference	Saman	Level 1
14		ADDED	Analytical	(137.629784797, -41.		342958	(137.6297	(137.629784797, -41.		5/4/2017 15:56	Hub	Saman	Level 1
15		ADDED	Analytical	(137.629784797, -41.		342959	(137.6297	(137.629784797, -41.		5/4/2017 15:56	Reference	Saman	Level 1
16		MODIFIED	Structural	(137.6297	2X2	342579	(138.6297	(136.629784797, -75.		5/4/2017 15:58	FamilyInst	Saman	South

Figure 5-10 Element Change CSV File

	A	B	C	D
		ErrorType	Time	User
0		Delete Selection	4/25/2017 12:35	Saman
1		Drag	4/25/2017 12:36	Saman
2		Drag	4/25/2017 12:36	Saman
3		Finish sketch	4/27/2017 16:37	Saman
4		Finish sketch	4/27/2017 16:38	Saman
5		Finish sketch	4/27/2017 16:38	Saman
6		Finish sketch	4/27/2017 16:38	Saman
7		Modify element attributes	4/27/2017 16:40	Saman
8		Window	4/27/2017 16:50	Saman
9		Structural Column	5/4/2017 13:05	Saman
10		Structural Column	5/4/2017 13:05	Saman
11		Structural Column	5/4/2017 13:05	Saman

Figure 5-11 Error CSV File

5.3.2 Experiment Results

The demographic information of the experiment participants is presented in Table 5-2 Demographic Information of Experiment Participants. Two modelers were beginners with little experience working with Revit prior to this experiment, while the other three modelers had done capstone building modeling projects with Revit prior to this experiment, and they were working as BIM modelers when the experiment was conducted. The experiment evaluated different sets of measures for building modeling activities that participants performed, and the calculated measures are reported anonymously to protect the participants' identities. Some preliminary measures regarding the modelers are reported in Table 5-3. The most frequent individual commands were largely executed to add elements such as walls and windows. The less experienced modelers also used the sketch mode of Revit relatively more frequently, and they often executed the "Finish_Sketch" command. All participants executed most commands using the buttons on the main top ribbon of Revit.

Table 5-2 Demographic Information of Experiment Participants

	Experience with Revit	**Age**	**Gender**
Modeler 1	Beginner (0-1 years)	20-25	Female
Modeler 2	Beginner (0-1 years)	20-25	Male
Modeler 3	Intermediate (1-3 years)	20-25	Female
Modeler 4	Intermediate (1-3 years)	20-25	Female
Modeler 5	Intermediate (1-3 years)	20-25	Male

Table 5-4 lists some information regarding software failures and the frequent view types that the experiment participants used. The total number of entries for novice modelers

is higher than for the others by a factor of at least 3. The plugin also recorded more errors for less experienced modelers (1 and 2) compared to the others. This observation can be expected, since novice Revit users are prone to make more mistakes. Additionally, since those two modelers used the sketch mode more often, the "Finish sketch" error was raised more frequently during their modeling sessions. Another common type of error was "Drag," which is raised when elements are moved to locations where there are geometric conflicts, for instance, when moving a door to a location on a model where a window already exists. "Join" errors occur when elements such as two walls or a wall and roof are not properly attached, and "Structural" errors also occur when the load bearing and non-bearing elements are attached and need to be separated (Autodesk 2017b).

Table 5-3 Preliminary Measures of Experiment Participants

	Frequent Individual Commands	**Command Execution Points**
Modeler 1	ID_FINISH_SKETCH ID_VIEW_DEFAULT_3D_VIEW ID_OBJECTS_WALL	Autodesk.Windows.RibbonButton Autodesk.Windows.RibbonSplitButton
Modeler 2	ID_FINISH_SKETCH ID_OBJECTS_WALL ID_OBJECTS_WINDOW	Autodesk.Windows.RibbonButton Autodesk.Windows.RibbonSplitButton
Modeler 3	ID_OBJECTS_WALL ID_VIEW_DEFAULT_3D_VIEW ID_OBJECTS_WINDOW	Autodesk.Windows.RibbonButton Autodesk.Windows.RibbonSplitButton
Modeler 4	ID_OBJECTS_WALL ID_OBJECTS_ROOM ID_OBJECTS_WINDOW	Autodesk.Windows.RibbonButton Autodesk.Windows.RibbonSplitButton
Modeler 5	ID_EDIT_MOVE ID_OBJECTS_WALL ID_OBJECTS_WINDOW	Autodesk.Windows.RibbonButton Autodesk.Windows.RibbonSplitButton

Table 5-4 Most Frequent Error and View Types

	Total Number of Entries	Total Number of Errors	Top Error Type	Type View Type
Modeler 1	17,318	162	Finish sketch Structural Column Drag	Level 1 {3D} East
Modeler 2	12,220	155	Drag Wall End Finish sketch Wall - Line	Level 1 {3D} East
Modeler 3	4,131	75	Join Walls to Roof Wall - Line Drag	Level 1 {3D} North
Modeler 4	4,740	61	Trim/Extend to Corner Wall - Line Drag	Level 1 {3D} Roof
Modeler 5	4,379	55	Wall - Line Select the roofs Finish sketch	Level 1 {3D} East

As previously specified, the experiment focused particularly on activities conducted to model walls and windows. Figure 5-12 illustrates the average number of commands that each modeler executed to model a wall or a window. Overall, the average number of modifications are higher for less experienced modelers compared to the more experienced ones. This can be expected, since expert modelers need fewer tries to obtain satisfactory results. Also, the modelers generally executed more commands to model walls compared to windows. However, this is not the case for modeler 4, and it can be attributed to the high number of drag operations that the user executed to place windows on their correct coordinates.

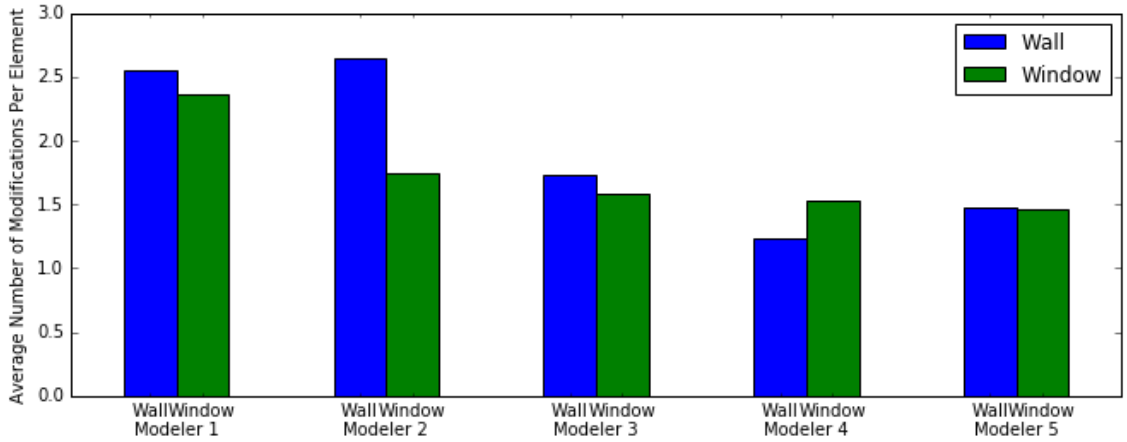

Figure 5-12 Wall and Window Modifications Bar Chart

The average time it took experiment participants to model a wall or window is illustrated in Figure 5-13. In calculating these values, the time difference between two consecutive commands corresponding to an element is considered to be 3 minutes if the actual difference is more than that. This assumption was made per a suggestion by the collaborating firm's designers to avoid counting idle times in measurements. Similar to the trend observed with the average number of modifications, novice modelers require more time to produce satisfactory building model elements. Overall, it takes modelers more time to design walls compared to windows.

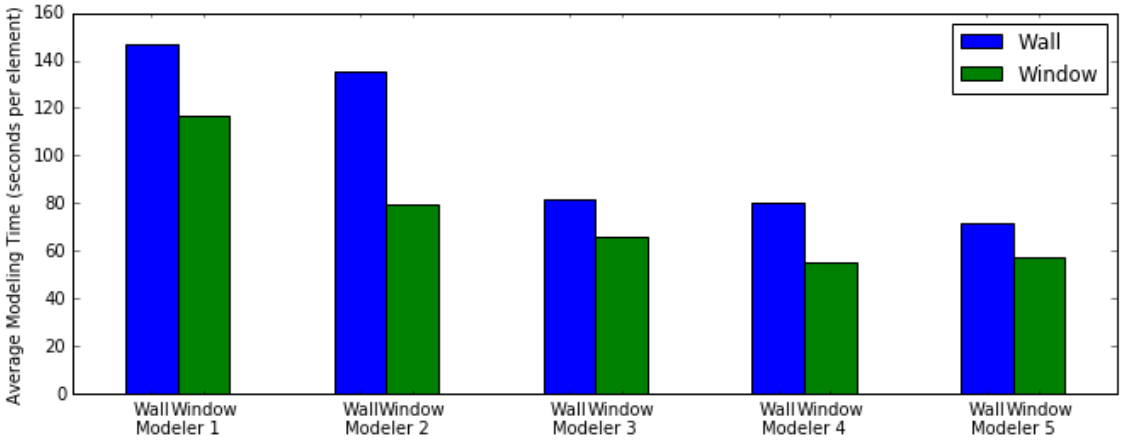

Figure 5-13 Wall and Windows Average Modeling Times

The relation between the average number of modifications per element and the average element design times is illustrated in Figure 5-14. As the trend lines depict, there is a direct correlation between these two variables. In general, it can be said that a higher number of modifications is associated with a higher average modeling time for walls and windows.

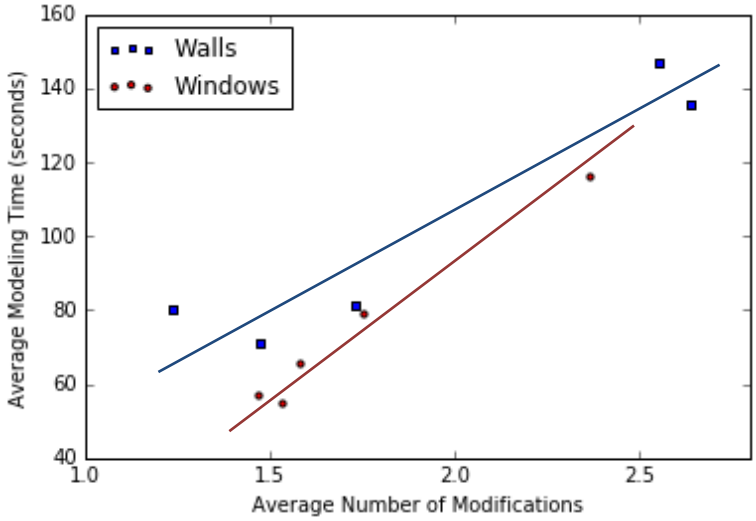

Figure 5-14 Average Modeling Times vs. Average Number of Modifications

5.3.3 Optimal Modeling Team Configuration

This section focuses on using individual production rates calculated based on plugin data to identify the optimal configuration of modeling teams. The proposed approach utilizes EDD sequencing in combination with the CPM. The EDD approach sequences jobs in their increasing order of due dates and thus minimizes maximum lateness, which is defined as follows (Baker & Trietsch 2013):

- Lateness (L_j) is the amount of time by which the completion time of job j exceeds its due date: $L_j = C_j - D_j$.

- Maximum lateness is calculated as $L_j = \max_{1 \le j \le n} \{L_j\}$.

In the context of this chapter, a "job" is defined as a set of building segments that are modeled separately, for example, the floors of a multistory building or building zones that a design team defines prior to the start of the modeling process. Each job consists of multiple design modeling tasks such as the modeling of structural, electrical, mechanical or architectural elements.

The amount of time required to finish each job is assessed using the CPM. First, available modelers are assigned to the given tasks based on their expertise. Next, the time it takes each modeler to finish his or her tasks is estimated based on both the number of elements that need to be modeled and the previously calculated individual production rates. The total time it takes to finish a job is then calculated using the CPM. Finally, the optimal configuration of the design modeling team is identified using the EDD approach. Predetermined due dates, which the design managers set, can be used to calculate the lateness criterion.

Figure 5-15 Framework of HypotheticalFigure 5-15 depicts the framework of a hypothetical design modeling project used to establish the functionality of the proposed method. The process consists of modeling walls and windows in three building zones. The design team consists of two modelers: one in charge of modeling walls, and one for modeling windows. The proposed approach and the individual production rates that were obtained through the experiment were used to identify the optimal assignment of modelers.

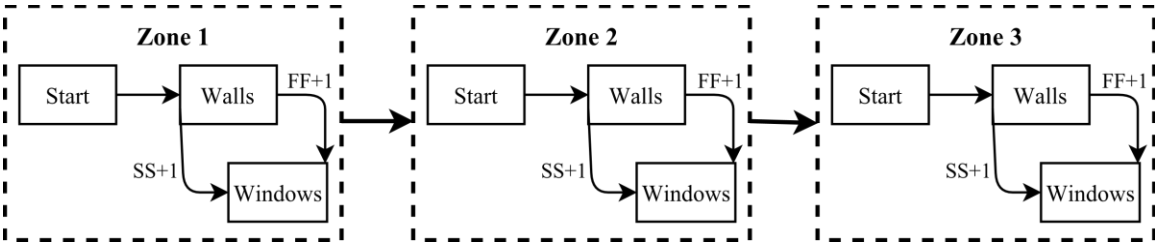

Figure 5-15 Framework of Hypothetical Project

The general characteristics of the three building zones in the proposed project are outlined in Table 5-5. These characteristics include the number of wall and window elements to be modeled as well as the predefined due dates for each zone. The amount of time that each modeler would need to model different elements was calculated by multiplying the number of elements specified in Table 5-5 by the individual production rates obtained from the experiment. The results were converted to 8-hour workdays based on an 80% productivity rate (Sacks & Barack 2008), as presented in Table 5-6. Due to their higher performance levels, only modelers 3, 4, and 5 were considered in the analysis. Similarly, in real-world scenarios, calculating individual production rates would allow managers to identify modelers with higher performance levels among the available candidates.

Table 5-5 General Parameters of Hypothetical Problem

	Zone 1	**Zone 2**	**Zone 3**
Total number of wall elements	1,500	800	1,150
Total number of window elements	700	1,200	800
Due date (days)	5	8	13

Table 5-6 Element Modeling Times Based on Production Rates

		Modeler 3	Modeler 4	Modeler 5
Individual production rate for a wall (hour)		0.023	0.022	0.02
Individual production rate for a window (hour)		0.018	0.015	0.016
Zone 1	Time required to model walls (days)	6	6	5
	Time required to model windows (days)	2	2	2
Zone 2	Time required to model walls (days)	3	3	3
	Time required to model windows (days)	4	3	3
Zone 3	Time required to model walls (days)	5	4	4
	Time required to model windows (days)	3	2	2

Given the number of available modelers, there are six possible configurations for the design team. The possible modeling task assignments for different configurations are listed in Table 5-7. In each configuration, one of the modelers is tasked with modeling walls, while the other models windows.

Table 5-7 Possible Team Configurations

	Modeler 3	Modeler 4	Modeler 5
Configuration 1	walls	windows	
Configuration 2	walls		windows
Configuration 3	windows	walls	
Configuration 4	windows		walls
Configuration 5		walls	windows
Configuration 6		windows	walls

The amount of time it would take different design team configurations to model each zone was calculated using the CPM (Table 5-8), whose dependencies are illustrated in Figure 5-15. Once the zone processing times were estimated, the completion times were calculated by scheduling the zones (jobs) in EDD order (i.e., earliest due date first), and each job's lateness was measured by comparing its completion time and due date. The maximum lateness values are listed in the last column of Table 5-9. The results indicate

that the optimal team arrangement is configuration 6, ($L_{max} = 2$), where modelers 4 and 5 would design windows and walls respectively.

Table 5-8 Zone Processing Times Based on the CPM

	Zone 1	Zone 2	Zone 3	Processing Times (days)
Configuration 1	7	4	6	
Configuration 2	7	4	6	
Configuration 3	7	4	5	
Configuration 4	6	5	5	
Configuration 5	7	4	5	
Configuration 6	6	4	5	

Table 5-9 Computation of Maximum Lateness for Different Team Configurations

		EDD Sequence			Maximum Lateness (L_{max})
		Zone 1	Zone 2	Zone 3	
Due date D_j		5	8	13	
Configuration 1	Completion time C_j	7	11	17	4
	Lateness L_j	2	3	4	
Configuration 2	Completion time C_j	7	11	17	4
	Lateness L_j	2	3	4	
Configuration 3	Completion time C_j	7	11	16	3
	Lateness L_j	2	3	3	
Configuration 4	Completion time C_j	6	11	16	3
	Lateness L_j	1	3	3	
Configuration 5	Completion time C_j	7	11	16	3
	Lateness L_j	2	3	3	
Configuration 6	Completion time C_j	6	10	15	2
	Lateness L_j	1	2	2	

5.4 Discussion of Results

As mentioned in the introduction section, the accurate measurement of design modeling performance metrics requires the automatic collection of model development data. The conducted experiment demonstrated that the developed Revit plugin is fully

capable of recording several modeling events, such as element changes, command executions, and errors, in real time. Unlike design log files, there was no need to use a text parser to gather the necessary information for analysis. The developed plugin also captured additional information, such as element GUID, coordination, family, type, and command execution points. These information items are not recorded in design log files and hence cannot be retrieved from them.

The LoD in the modeled project was low, thereby limiting the collected data to basic wall, window, door, ceiling, and roof elements. It is likely that increasing the LoD will result in higher values for the average number of modifications and average modeling time per elements. This conclusion is justified because more effort is required to model elements with higher LODs, given their complexity. Another important observation is that the recorded data captured the variations that exist among all modelers (novice and experienced) in terms of the time it takes them to model different elements. In fact, to create the wall and window elements, novice modelers needed approximately 84% and 48% more time respectively than experienced modelers. Therefore, this methodology can be used in design offices to develop performance profiles for modelers based on their work in developing different building systems in multiple projects.

The researcher tested the quality of 3D models that students developed. Furthermore, the average number of element modifications was used as a measure to assess the quality of building models. If the average number of modifications by a user is significantly higher than that of his peers, then it may indicate that he or she requires more attempts to create a model of comparable quality. This trend was observed in the obtained results as novice modelers needed at least 50% and 41% more modifications on average

for wall and window elements respectively. Another possible method for assessing model quality is to compare the average number of modified elements after clash detection sessions. A statistically significant higher number of modified elements may indicate a need for adding elements or rerouting too many of them. This could be due to factors such as low experience, the utilization of substandard elements, or having an incomplete model. The validity of this quality assessment methodology should be tested using large model development events and clash detection datasets that are collected from a team of modelers working on different building systems.

The metrics that were measured through the conducted experiment were used to find the optimal modeling team configuration in a hypothetical project. The required calculations did not necessitate extensive effort, given the relatively small size of the participants and the simplicity of the hypothetical project. It should be noted that this exercise was performed to illustrate the application of the proposed approach, and no statistically significant conclusion can be drawn from it. However, in full-scale industry projects, the amount of required calculations will be substantially higher. This increase is due to the large number of modeling tasks as well as the high number of possible configurations in real-world projects. Therefore, it is recommended that scheduling software solutions, such as Microsoft Project, be used for industry projects to calculate processing times for available jobs. Also, optimization platforms such as ILOG CPLEX can be utilized to evaluate possible team configurations and find the optimal one.

5.5 Chapter Summary

This chapter outlined the details of an experiment that was conducted to both capture design modeling performance data and utilize the obtained information to identify the optimal modeling team configuration. Five graduate students from the Georgia Institute of Technology's School of Architecture were selected for this experiment. The participants used the plans of a youth and family center building to produce 3D models on Revit. Meanwhile, the plugin introduced in the previous chapter captured their model development data. The collected data were then analyzed using Python's pandas data analysis library. The pandas data frames were used to efficiently store data and to calculate the individual modeling performance measures.

Individual production rates from the experiment participants were used to establish the validity of an approach proposed to identify optimal design team configurations. The presented approach uses the EDD sequencing rule in combination with the CPM to calculate the maximum lateness for different design team arrangements. The arrangement with the smallest maximum lateness value was selected as the optimal modeling team configuration.

This section contributes to the state of knowledge by creating a mathematical model to estimate design modeling project completion times based on individual performance data and project requirements. This chapter also contributes to the state of practice by enabling design managers to identify an optimal modeling team arrangement based on automatically captured quantitative performance information.

CHAPTER 6: CONCLUSIONS, LIMITATIONS, AND FUTURE DIRECTIONS

6.1 Introduction

As capital projects are becoming more complex, their design modeling processes increasingly require collaborative efforts among various AEC disciplines. Throughout this process, the different priorities of design modelers often result in conflicts that can negatively impact project outcomes (for example, a modeler falling behind the design schedule due to working on other projects). There is a need for the effective management of the modeling process to prevent such unwanted outcomes. However, a review of the existing literature demonstrates that the current methods of design management lack objective measurement systems to quantify performance in modeling. Additionally, existing methodologies rely on manually collected data that lack the accuracy required for correct measurements. The difficulties associated with evaluating design modeling performance renders the existing methodologies impractical. This research improves upon previous efforts by presenting a novel API-based approach to automatically collect granular design development data directly from modeling software packages and to efficiently calculate several modeling performance measures.

A comprehensive review of the existing body of knowledge regarding the design management practices and motivations behind this study was presented in chapter 2. The next chapter investigated the presence of frequent command execution patterns using the GST pattern mining algorithm. In chapter 4, an API-based, object-oriented data collection framework was introduced, and the steps that were conducted to implement the framework

as an Autodesk Revit plugin were outlined. Finally, chapter 5 describes the details of an approach that is proposed to identify the optimal design modeling team configuration based on quantitative performance data.

6.2 Summary of Results and Contributions to the Body of Knowledge

In chapter 3, the utilization of modeling development information, embedded in design log files that Autodesk Revit produces, was proposed as a rich source of performance data. To this end, the necessary steps to extract and analyze the data were outlined. Generalized suffix tree data structures were utilized to find common command sequences among Revit users. Frequent command patterns were identified by conducting a DFS on constructed GST trees, and the extracted patterns for different users were compared against each other to identify shared sequences. In addition to identifying the common command execution patterns, the average time it takes the selected modelers to execute command sequences was calculated. The obtained results demonstrate that there is a statistically significant difference among the modelers in terms of the time it takes them to conduct similar modeling tasks. This chapter contributes to the state of knowledge by proposing a tailored string mining algorithm that is capable of extracting meaningful information from timestamped design development data. The proposed methodology contributes to the state of practice by enabling design project managers to gain unprecedented insight into the evolution of a building model using the information embedded in design log files.

Chapter 4 utilized modeling software solution's APIs to automatically collect and store timestamped design development information. The proposed passive data recording approach allows for the real-time capture of comprehensive UI interactions and model element modification events. These recordings consist of all possible software actions, ranging from creating, selecting, and modifying elements to navigating through different zones of a virtual model. Such recordings, constituting timestamped event sequences, are organized and stored in machine-readable .txt file formats. The proposed framework was also implemented as an Autodesk Revit plugin, and an experiment was conducted to verify the accuracy of this plugin. Throughout this experiment, manual recordings of model development events were compared against the automatically generated plugin output. This section contributes to the state of knowledge by introducing a framework to automatically capture accurate performance data from design software solutions, and it contributes to the state of practice by helping design managers to acquire progress information without the need to manually record and report data.

Chapter 5 outlined the details of an approach to identify the optimal design modeling team configuration based on automatically collected performance data. To this end, an experiment was conducted to capture data using the Revit plugin introduced in the previous chapter. The collected data were then analyzed using Python's pandas data analysis library to produce individual performance measures. The experiment participants' individual production rates were used to establish the validity of an approach proposed to identify the optimal design team configurations. The presented approach uses the EDD sequencing rule in combination with the CPM to calculate the maximum lateness for different design team arrangements. This section contributes to the state of knowledge by

creating a mathematical model to estimate design modeling project completion times based on individual performance data and project requirements. The presented model can be expanded to include other decision criteria as software has more data capture capabilities, in addition to time stamps. This chapter also contributes to the state of practice by enabling design managers to identify an optimal modeling team arrangement based on automatically captured quantitative performance information.

6.3 Limitations of the Current Study

The results of this study should be interpreted considering the following limitations:

- The analysis presented in chapter 3 focused on modelers working on interior systems of healthcare projects, and the findings of the chapter should not be generalized to other building systems, trades, and projects. Moreover, no statistical significance was sought from the analysis, given the relatively small size of the data set. There is a need for a larger sample of design development data from various trades and projects to generalize the obtained results.
- The experiment conducted in chapter 5 focused on collecting data to illustrate the workings of the proposed optimal modeling team configuration methodology. Given the small number of participants, no statistically significant conclusion was drawn from the results of this experiment. Therefore, the findings of this chapter are subject to validation from a larger sample and a robust statistical study.

- When analyzing the design development data in chapter 3, it was assumed that the developed building models meet the same quality criteria. This assumption was made since the quality and LoD-level information for these models were not provided to the author. Moreover, the building models that all modelers created had gone through a quality control process, which the collaborating company established. However, it is recommended that future studies should measure and consider the quality of the information in their analyses to attain generalizable findings.

6.4 Future Works and Directions

The primary aim of this dissertation was to develop a methodology for automatically capturing design development data for modeling performance measurement. Additionally, an approach was presented to identify optimal design team configurations based on performance measurements. Future research could concentrate on the following directions:

- The fine-grained information captured from design modeling software solutions could be used to develop customized training programs. Furthermore, API-based data collection systems enable managers to collect granular data on the actions of each designer during modeling sessions, and such a wealth of information can be used to benchmark modelers' performance in their areas of expertise. These detailed benchmarks allow designers to identify their weaknesses for further improvement. For instance, an architectural modeler can

compare his or her performance in designing a component, such as curved or spiral stairs, with that of his or her peers. If the modeler is comparatively underperforming, the project manager can provide customized training to improve his or her performance in designing the component.

- The granular modeling performance data can be used for developing advanced, real-time, visualization dashboards, allowing project managers to summon the relevant context when faced with a modeling conflict. Often, too little information is provided to convey a message about an actual design modeling problem, making it problematic for project managers to track issues and determine the other design disciplines that will be affected by the problem. A multilayered visualization platform can support design managers to quickly and easily identify design issues, analyze the causes of these design problems, and communicate the problem to the design team at the selected level of detail.
- The real-time monitoring of design development events could enable managers to detect and prevent poor modeling practices. The main purpose of BIM tools is to generate digital representations of the physical and functional characteristics of capital projects. Therefore, these virtual models should include accurate information regarding the actual size, location, and other properties of each element. However, in practice, some models lack the intended quality. For instance, a poor modeling practice that usually remains undetected is the excessive utilization of imported substandard families in virtual models. Some of these families—often created by manufacturers to model their products—are corrupted, contain too much detail, or miss important

requirements such as parametric information. Such models increase the file size, and in extreme cases, this results in degrading software performance. Given the substantial number of elements in a typical virtual model, it is impossible to manually identify these poor practices. The API-based monitoring of virtual design models enables managers to detect such damaging practices and prevent them from leading to further issues in other parts of the system.

- The wide range of information collected using design solution APIs can help software developers to improve a user's experience. The command patterns that are identified for modelers could be used to create shortcut keys that conduct a sequence of tasks at once. Additionally, information regarding mouse movements and the most frequently executed commands may be utilized to improve software UI designs.

REFERENCES

Abdul-Rahman, H. (1995). The cost of non-conformance during a highway project: a case study. Construction Management and Economics, 13(1), 23-32. DOI: http://dx.doi.org/10.1080/01446199500000004

Alzraiee, H., Moselhi, O., & Zayed, T. (2012). Dynamic planning of earthmoving projects using system dynamics. Gerontechnology, 11(2), 316. URL: https://pdfs.semanticscholar.org/e9cb/f6050f50b640e67c0325a676ca56a6a0d5ed.pdf

American Institute of Architects (2013). Project Building Information Modeling protocol form, URL: http://aiad8.prod.acquia-sites.com/sites/default/files/2016-09/AIA-G202-2013-Free-Sample-Preview.pdf

Anumba, C., Ugwu, O. O., Newnham, L., & Thorpe, A. (2002). Collaborative design of structures using intelligent agents. Automation in construction, 11(1), 89-103. DOI: https://doi.org/10.1016/S0926-5805(01)00055-3

Ashuri, B., Yarmohammadi, S., & Shahandashti, M. (2014). A critical review of methods used to determine productivity of mechanical, electrical, and plumbing systems coordination. In Construction Research Congress 2014: Construction in a Global Network, 777-786, DOI: https://doi.org/10.1061/9780784413517.080

Autodesk (2016). Accessed on: https://knowledge.autodesk.com/support/revit-products/getting started/caas/CloudHelp/cloudhelp/2015/ENU/Revit-GetStarted/files/GUID-477C6854-2724-4B5D-8B95-9657B636C48D-htm.html

Autodesk (2017a). Accessed on: https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2016/ENU/Revit-API/files/GUID-CEF0F9C9-046E-46E2-9535-3B9620D8A170-htm.html

Autodesk (2017b). Accessed on: https://knowledge.autodesk.com/support/revit-products/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Revit-Troubleshooting/files/GUID-F0945713-4389-4F8E-B5DB-DCE03A8C1ADF-htm.html

Baars, H., & Kemper, H. G. (2008). Management support with structured and unstructured data—an integrated business intelligence framework. Information Systems Management, 25(2), 132-148. DOI: http://dx.doi.org/10.1080/10580530801941058

Baker, K. R., & Trietsch, D. (2013). Principles of sequencing and scheduling. John Wiley & Sons, Inc. New York.

Bassioni, H.A., Price, A.D.F. & Hassan, T.M. (2005), Building a conceptual framework for measuring business performance in construction: an empirical evaluation, Construction Management and Economics, No. 5, 495-507. DOI: http://dx.doi.org/10.1080/0144619042000301401

Bate, P., & Robert, G. (2007). Bringing user experience to healthcare improvement: The concepts, methods and practices of experience-based design. Radcliffe Publishing.

Benayoune, M., & McGreavy, C. (1994). Concurrent engineering system for process design and operation. In Institution of Chemical Engineers Symposium Series.

Boton, C., Halin, G., Kubicki, S., & Forgues, D. (2015). Challenges of big data in the age of building information modeling: a high-level conceptual pipeline. In International Conference on Cooperative Design, Visualization and Engineering, 48-56. DOI: 10.1007/978-3-319-24132-6_6

Caldas, C. H., Soibelman, L., & Han, J. (2002). Automated classification of construction project documents. Journal of Computing in Civil Engineering, 16(4), 234-243. DOI: http://dx.doi.org/10.1061/(ASCE)0887-3801(2002)16:4(234)

Caldas, C. H., & Soibelman, L. (2003). Automating hierarchical document classification for construction management information systems. Automation in Construction, 12(4), 395-406. DOI: http://dx.doi.org/10.1016/S0926-5805(03)00004-9

Caldas, C. H., Soibelman, L., & Gasser, L. (2005). Methodology for the integration of project documents in model-based information systems. Journal of Computing in Civil Engineering, 19(1), 25-33. DOI: http://dx.doi.org/10.1061/(ASCE)0887-3801(2005)19:1(25)

Chalabi, A. F., Beaudin, B. J., & Salazar, G. F. (1987). Input variables impacting design effectiveness. Construction Industry Institute, University of Texas at Austin.

Chang, A. S., & Ibbs, W. (2006). System model for analyzing design productivity. Journal of Management in Engineering, 22(1), 27-34. DOI: https://doi.org/10.1061/(ASCE)0742-597X(2006)22:1(27)

Chiu, C. Y., & Russell, A. D. (2011). Design of a construction management data visualization environment: A top–down approach. Automation in Construction, 20(4), 399-417. Doi: https://doi.org/10.1016/j.autcon.2010.11.010

Construction Industry Institute (CII). (2001). “Engineering productivity measurement.” RR156-11, Construction Industry Institute, Univ. of Texas at Austin, Austin, TX.

Construction Industry Institute (CII). (2004). “Engineering productivity measurements II.” RR192-11, Construction Industry Institute, Univ. of Texas at Austin, Austin, TX.

Dave, B., Kubler, S., Främling, K., & Koskela, L. (2016). Opportunities for enhanced lean construction management using Internet of Things standards. Automation in Construction, 61, 86-97. DOI: https://doi.org/10.1016/j.autcon.2015.10.009

De Marco, A., & Narbaev, T. (2013). Earned value-based performance monitoring of facility construction projects. Journal of Facilities Management, 11 Issue: 1, 69-80. DOI: https://doi.org/10.1108/14725961311301475

Ding, Z., Ng, F., & Li, J. (2014). A parallel multiple mediator model of knowledge sharing in architectural design project teams. International Journal of Project Management, 32(1), 54-65. DOI: http://dx.doi.org/10.1016/j.ijproman.2013.04.004

Doloi, H. (2012). Cost overruns and failure in project management: Understanding the roles of key stakeholders in construction projects. Journal of construction engineering and management, 139(3), 267-279. DOI: https://doi.org/10.1061/(ASCE)CO.1943-7862.0000621

Du, J., Liu, R., & Issa, R. R. (2014). BIM cloud score: benchmarking BIM performance. Journal of Construction Engineering and Management, 140(11), 04014054. DOI: https://doi.org/10.1061/(ASCE)CO.1943-7862.0000891

Eastman, C., Eastman, C. M., Teicholz, P., & Sacks, R. (2011). BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors. John Wiley & Sons 2nd edition.

Egan, S.J. (1998). Rethinking Construction, the Report of the Construction Task Force on the Scope for Improving the Quality and Efficiency of the UK Construction Industry, Department of Environment, Transport and the Regions (DETR), London, UK.

Engineering News Record (ENR) (2013). Information mobility: improving team collaboration through movement of project information, SmartMarket report, McGraw-Hill Construction, New York. URL: http://enr.construction.com/engineering/pdf/News/Information_Mobility_SMR_2013.pdf

Evins, R. (2013). A review of computational optimization methods applied to sustainable building design. Renewable and Sustainable Energy Reviews, 22, 230-245. doi: http://dx.doi.org/10.1016/j.rser.2013.02.004

Fan, H., Xue, F., & Li, H. (2014). Project-Based as-needed information retrieval from unstructured AEC documents. Journal of Management in Engineering, 31(1). DOI: http://dx.doi.org/10.1061/(ASCE)ME.1943-5479.0000341

Fayek, A. R., & Sun, Z. (2001). A framework for evaluating design project performance A. Robinson Fayek & Z. Sun. Creative Systems in Structural and Construction Engineering, Balkema, Netherlands, 191.

Girard, P., & Robin, V. (2006). Analysis of collaboration for project design management. Computers in industry, 57(8), 817-826. DOI: https://doi.org/10.1016/j.compind.2006.04.016

Gautam, G., & Yadav, D. (2014). Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In Contemporary computing (IC3), 437-442, IEEE. DOI: https://doi.org/10.1109/IC3.2014.6897213

Girczyc, E., & Carlson, S. (1993). Increasing design quality and engineering productivity through design reuse. In Proceedings of the 30th international Design Automation Conference, 48-53, ACM, DOI: https://doi.org/10.1145/157485.164565

Gog, S., Beller, T., Moffat, A., & Petri, M. (2014). From theory to practice: Plug and play with succinct data structures. In International Symposium on Experimental Algorithms, 326-337. DOI: http://dx.doi.org/10.1007/978-3-319-07959-2_28

Graham, B. B. (1990). Applying software tools to enhance engineering group productivity. Proceedings of the 5th annual Applied Power Electronics Conference and Exposition, APEC'90, 612-618, IEEE, DOI: https://doi.org/10.1109/APEC.1990.66360

Gu, N., & London, K. (2010). Understanding and facilitating BIM adoption in the AEC industry. Automation in construction, 19(8), 988-999. DOI: https://doi.org/10.1016/j.autcon.2010.09.002

Gusfield, D. (1997). Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge university press.

Guerbas, A., Addam, O., Zaarour, O., Nagi, M., Elhajj, A., Ridley, M., & Alhajj, R. (2013). Effective web log mining and online navigational pattern prediction. Knowledge-Based Systems, 49, 50-62. DOI: http://dx.doi.org/10.1016/j.knosys.2013.04.014

Hinze, J., Thurman, S., & Wehle, A. (2013). Leading indicators of construction safety performance. Safety Science, 51(1), 23-28. DOI: https://doi.org/10.1016/j.ssci.2012.05.016

Isbell, T.S. (1993). Concurrent engineering planning in HGST systems. High speed ground transportation systems.

Jansson, G., Viklund, E., & Lidelöw, H. (2016). Design management using knowledge innovation and visual planning. Automation in Construction, 72, 330-337. DOI: https://doi.org/10.1016/j.autcon.2016.08.040

Jin, Z., Deng, F., Li, H., & Skitmore, M. (2013). Practical framework for measuring performance of international construction firms. Journal of Construction Engineering and Management, 139(9), 1154-1167. DOI: http://dx.doi.org/10.1061/(ASCE)CO.1943-7862.0000718

Kim, I. (2007). "Development and implementation of an engineering productivity measurement system (EPMS) for benchmarking." Ph.D. dissertation, Univ. of Texas at Austin, Austin, TX.

Knotten, V., Svalestuen, F., (2014). Implementing Virtual Design and Construction (VDC) in Veidekke - Using Simple Metrics to Improve the Design Management Process. In: Proceedings of the 22nd Annual Conference of the International Group for Lean Construction, 3, Akademika Forlag, 1379-1390.

Korman, T., Simonian, L., & Speidel, E. (2008). Using Building Information Modeling to improve the mechanical, electrical, and plumbing coordination process for buildings. In Proceedings of the AEI 2008 Conference, Colorado, USA. DOI: https://doi.org/10.1061/41002(328)10

Kymmell, W (2008). Building information modeling: planning and managing construction projects with 4D CAD and simulations, New York: McGraw Hill. URL: https://www.accessengineeringlibrary.com/browse/building-information-modeling-planning-and-managing-construction-projects-with-4d-cad-and-simulations-mcgraw-hill-construction-series

Lee, G. & Kim J. W. (2014). Parallel vs. Sequential Cascading MEP Coordination Strategies: A Pharmaceutical Building Case Study. Automation in Construction, 43, 170-179. DOI: https://doi.org/10.1016/j.autcon.2014.03.004

Lee, S., & Peña-Mora, F. (2007). Understanding and managing iterative error and change cycles in construction, System Dynamics Review, 23(1), 35-60. DOI: https://doi.org/10.1002/sdr.359

Leite, F., Akcamete, A., Akinci, B., Atasoy, G., & Kiziltas, S. (2011). Analysis of modeling effort and impact of different levels of detail in building information models. Automation in Construction, 20(5), 601-609. DOI: https://doi.org/10.1016/j.autcon.2010.11.027

Leong, M. S., & Tilley, P. (2008). A lean strategy to performance measurement–reducing waste by measuring next customer needs. In Proceedings for the 16th Annual

Conference of the International Group for Lean Construction Safety, Quality and the Environment, 757-768. University of Salford.

Liao, P. C., Thomas, S. R., O'Brien, W. J., Mulva, S. P., & Dai, J. (2009). Development of project level engineering productivity benchmarking index, In Construction Research Congress 2009: Building a Sustainable Future, 1087-1095. DOI: https://doi.org/10.1061/41020(339)110

Liao, P. C., O'Brien, W. J., Thomas, S. R., Dai, J., & Mulva, S. P. (2011). Factors affecting engineering productivity, Journal of Management in Engineering, 27(4), 229-235. DOI: http://dx.doi.org/10.1061/(ASCE)ME.1943-5479.0000059

Love, P. E., & Li, H. (2000). Quantifying the causes and costs of rework in construction. Construction Management & Economics, 18(4), 479-490. doi: http://dx.doi.org/10.1080/01446190050024897

Love, P. E., Edwards, D. J., Irani, Z., & Forcada, N. (2014). The latent causes of rework in floating production storage and offloading projects, Journal of Civil Engineering and Management, 20(3), 315-329. DOI: http://dx.doi.org/10.3846/13923730.2013.802725

Lu, W., Fung, A., Peng, Y., Liang, C., & Rowlinson, S. (2014). Cost-benefit analysis of Building Information Modeling implementation in building projects through demystification of time-effort distribution curves, Building and Environment, 82, 317-327. DOI: https://doi.org/10.1016/j.buildenv.2014.08.030

Manning, R, & Messner, J. (2008). Case studies in BIM implementation for programming of healthcare facilities, Journal of Information Technology in Construction, Special Issue Case studies of BIM use, 246-257, http://www.itcon.org/2008/18

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. URL:http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation

McGeorge, J. F. (1988). Design productivity: a quality problem. Journal of Management in Engineering, 4(4), 350-362. DOI: https://doi.org/10.1061/(ASCE)9742-597X(1988)4:4(350)

McGraw-Hill Construction (2013). Lean Construction Leveraging Collaboration and Advanced Practices to Increase Project Efficiency, SmartMaket report, McGraw-Hill, New York. URL: http://www.leanconstruction.org/media/docs/Lean_Construction_SMR_2013.pdf

Meredith J. & Mantel S. (2003). Project management: a managerial approach. J. Wiley and Sons, Fifth Ed.

Navon, R., & Sacks, R. (2007). Assessing research issues in automated project performance control (APPC). Automation in Construction, 16(4), 474-484. DOI: https://doi.org/10.1016/j.autcon.2006.08.001

Pandas Library, URL: http://pandas.pydata.org/

Park, C. S., Lee, D. Y., Kwon, O. S., & Wang, X. (2013). A framework for proactive construction defect management using BIM, augmented reality and ontology-based

data collection template. Automation in Construction, 33, 61-71. DOI: https://doi.org/10.1016/j.autcon.2012.09.010

Pei, J., Han, J., & Lakshmanan, L. V. (2001a). Mining frequent itemsets with convertible constraints. Proceedings. 17th International Conference on Data Engineering. 433-442). IEEE. DOI: https://doi.org/10.1109/ICDE.2001.914856

Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M-C (2001b). PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. Proceeding of the 2001 International Conference on Data Engineering (ICDE'01), Heidelberg, Germany, 215–224. URL: http://jayurbain.com/msoe/cs498-datamining/prefixspan_mining_sequential_patterns_by_prefix_projected_growth.pdf

Pilehchian, B., Staub-French, S., & Nepal, M. P. (2015). A conceptual approach to track design changes within a multi-disciplinary building information modeling environment. Canadian Journal of Civil Engineering, 42(2), 139-152. DOI: https://doi.org/10.1139/cjce-2014-0078

Plume, J., & Mitchell, J. (2007). Collaborative design using a shared IFC building model—Learning from experience. Automation in Construction, 16(1), 28-36. DOI: https://doi.org/10.1016/j.autcon.2005.10.003

Ren, Z., Anumba, C. J., & Yang, F. (2013). Development of CDPM matrix for the measurement of collaborative design performance in construction. Automation in Construction, 32, 14-23. DOI: https://doi.org/10.1016/j.autcon.2012.11.019

Revit API Documentation (2016). Accessed on: http://www.revitapidocs.com/2016/3c452286-57b1-40e2-2795-c90bff1fcec2.htm

Revit API Documentation (2017). Accessed on: https://knowledge.autodesk.com/support/revit-products/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Revit-Model/files/GUID-4EBB97AD-C7B6-4828-91EB-BC0E99B81E43-htm.html

Riley D. R., Varadan P., James J., & Thomas H. (2005). Benefit-cost metrics for design coordination of mechanical, electrical, and plumbing systems in multistory buildings, Journal of construction engineering and management, 131, 877-889. DOI: https://doi.org/10.1061/(ASCE)0733-9364(2005)131:8(877)

Sackett, P.J. & Evans, S. (1984). Computer Aided Engineering, productivity and quality of working life, Paper delivered at conference on Human Factors in Manufacturing, London, April.

Sacks, R., & Barak, R. (2008). Impact of three-dimensional parametric modeling of buildings on productivity in structural engineering practice, Automation in Construction, 17(4), 439-449. DOI: https://doi.org/10.1016/j.autcon.2007.08.003

Sacks, R., & Barak, R. (2010). Teaching Building Information Modeling as an Integral Part of Freshman Year Civil Engineering Education, Journal of Professional Issues in Engineering Education and Practice, 136(1), 30-38. DOI: https://doi.org/10.1061/(ASCE)EI.1943-5541.0000003

Sacks, R., Radosavljevic, M., & Barak, R. (2010). Requirements for building information modeling based lean production management systems for construction, Automation in construction, 19(5), 641-655. DOI: https://doi.org/10.1016/j.autcon.2010.02.010

Shahtaheri, M., Nasir, H., & Haas, C. T. (2014). Setting Baseline Rates for On-Site Work Categories in the Construction Industry, Journal of Construction Engineering and Management, 141(5). DOI: https://doi.org/10.1061/(ASCE)CO.1943-7862.0000959

Simpeh, E. K., Ndihokubwayo, R., Love, P. E., & Thwala, W. D. (2015). A rework probability model: a quantitative assessment of rework occurrence in construction projects, International Journal of Construction Management, 15(2), 109-116. DOI: http://dx.doi.org/10.1080/15623599.2015.1033814

Skibniewski, M. J., & Ghosh, S. (2009). Determination of key performance indicators with enterprise resource planning systems in engineering construction firms, Journal of Construction Engineering and Management, 135(10), 965-978. DOI: http://dx.doi.org/10.1061/(ASCE)0733-9364(2009)135:10(965)

Soibelman, L., & Kim, H. (2002). Data preparation process for construction knowledge generation through knowledge discovery in databases. Journal of Computing in Civil Engineering, 16(1), 39-48. DOI: http://dx.doi.org/10.1061/(ASCE)0887-3801(2002)16:1(39)

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: generalizations and performance improvements, In International Conference on Extending Database Technology, 1-17. DOI: 10.1007/BFb0014140

Staub-French S & Khanzode A (2007). 3D and 4D modeling for design and construction coordination: issues and lessons learned, Journal of Information Technology in Construction, 12, 381-407, http://www.itcon.org/2007/26

Succar, B., Sher, W., & Williams, A. (2012). Measuring BIM performance: Five metrics. Architectural Engineering and Design Management, 8(2), 120-142. DOI: http://dx.doi.org/10.1080/17452007.2012.659506

Sun, M., & Meng, X. (2009). Taxonomy for change causes and effects in construction projects. International Journal of Project Management, 27(6), 560-572. DOI: http://dx.doi.org/10.1016/j.ijproman.2008.10.005

Taylor-Adams, S., & Kirwan, B. (2013). Human reliability data requirements. Disaster Prevention and Management: An International Journal. DOI: http://dx.doi.org/10.1108/09653569710193754

Thomas, H. R., Korte, Q. C., Sanvido, V. E., & Parfitt, M. K. (1999). Conceptual model for measuring productivity of design and engineering, Journal of architectural engineering, 5(1), 1-7. DOI: https://doi.org/10.1061/(ASCE)1076-0431(1999)5:1(1)

Tizani, W. (2011). Collaborative Design in Virtual Environments at Conceptual Stage. In Intelligent Systems, Control and Automation: Science and Engineering, 48, 67-76. DOI: https://link.springer.com/chapter/10.1007/978-94-007-0605-7_6

Torbett R, Salter AJ, Gann DM, Hobday M (2001). Design performance measurement in the construction sector: a pilot study. University of Sussex, Brighton

Tucker, R.L. & Scarlett, B.R. (1986). Evaluation of design effectiveness, Construction Industry Institute. The University of Texas at Austin.

United States Census Burea. (2015). Construction Spending, URL: https://www.census.gov/construction/c30/c30index.html

Volk, R., Stengel, J., & Schultmann, F. (2014). Building Information Modeling (BIM) for existing buildings—Literature review and future needs. Automation in construction, 38, 109-127. DOI: https://doi.org/10.1016/j.autcon.2013.10.023

Verma, M., & Mehta, D. D. (2014). A Comparative study of Techniques in Data mining. International Journal of Emerging Technology and Advanced Engineering, 4(4), 314-321. URL: https://pdfs.semanticscholar.org/5709/2257696e87e154d566fccda2c70fb57db75e.pdf

Wang, X., & Tsai, J. J. H. (Eds.). (2011). Collaborative Design in Virtual Environments, 48, Springer Science & Business Media.

White, D., Sritharan, S., Suleiman, M., Mekkawy, M., & Chetlur, S. (2005). Identification of the best practices for design, construction, and repair of bridge approaches, No. CTRE Project 02-118. URL: http://www.ctre.iastate.edu/reports/tr481.pdf

Williams, T. P., & Gong, J. (2014). Predicting construction cost overruns using text mining, numerical data and ensemble classifiers, Automation in Construction, 43, 23-29. DOI: http://dx.doi.org/10.1016/j.autcon.2014.02.014

Winter, P. (1992). Computer aided process engineering: The evolution continues, Chemical Engineering Progress, 88(2), 76-83.

Xiao, Y., & Dunham, M. H. (2001). Efficient mining of traversal patterns. Data & Knowledge Engineering, 39(2), 191-214. DOI: http://dx.doi.org/10.1016/S0169-023X(01)00039-8

Yan X, Han J, Afshar R (2003). CloSpan: mining closed sequential patterns in large datasets, In: Proceeding of the 2003 SIAM international conference on data mining (SDM'03), San Francisco, CA, 166–177. DOI: http://dx.doi.org/10.1137/1.9781611972733.15

Yang, H., Yeung, J. F., Chan, A. P., Chiang, Y. H., & Chan, D. W. (2010). A critical review of performance measurement in construction", Journal of Facilities Management, 8 Issue: 4, 269-284. DOI: https://doi.org/10.1108/14725961011078981

Yarmohammadi, S., & Ashuri, B. (2015). Exploring the approaches in the implementation of BIM-based MEP coordination in the USA. Journal of Information Technology in Construction, 20, 347-363, http://www.itcon.org/2015/22

Yarmohammadi, S., Pourabolghasem, R., Shirazi, A., & Ashuri, B. (2016), A sequential pattern mining approach to extract information from BIM design log files, Proceedings of the International Symposium on Automation and Robotics in Construction (ISARC); Vilnius 33, 1-7, URL: http://search.proquest.com/openview/381e74c3bbf2fabb45142b8d0e350637/1?pq-origsite=gscholar&cbl=1646340

Yarmohammadi, S., Pourabolghasem, R., & Castro-Lacouture, D. (2017). Mining implicit 3D modeling patterns from unstructured temporal BIM log text data. Automation in Construction, 81, 17-24. DOI: https://doi.org/10.1016/j.autcon.2017.04.012

Yin, Y., Qin, S., & Holland, R. (2011). Development of a design performance measurement matrix for improving collaborative design during a design process, International Journal of Productivity and Performance Management, 60(2), 152-184. DOI: https://doi.org/10.1108/17410401111101485

Zhang, S., Teizer, J., Lee, J. K., Eastman, C. M., & Venugopal, M. (2013). Building information modeling (BIM) and safety: Automatic safety checking of construction models and schedules. Automation in Construction, 29, 183-195. DOI: https://doi.org/10.1016/j.autcon.2012.05.006